A PROGRAMOZÁS TANULÁSÁNAK ÉS TANÍTÁSÁNAK TÁMOGATÁSA ELEKTRONIKUS TANANYAGBA BEÉPÍTHETŐ INTERAKTÍV ANIMÁCIÓS MODELLEKKEL

DOKTORI ÉRTEKEZÉS

VÉGH LADISLAV

2017

Eötvös Loránd Tudományegyetem Informatika Doktori Iskola
Az informatika alapjai és módszertana doktori program

Iskolavezető: Dr. Csuhaj Varjú Erzsébet
az MTA doktora, egyetemi professzor

Programvezető: Dr. Demetrovics János
matematikai tudományok doktora, egyetemi professzor

Témavezetők:

Prof. Ing. Stoffa Veronika, CSc.
egyetemi professzor

Dr. Turcsányiné Szabó Márta
egyetemi docens

Média- és Oktatásinformatika Tanszék

Budapest, 2017

Köszönetnyilvánítás

Köszönöm témavezetőim, Prof. Ing. Stoffa Veronika, CSc. (Nagyszombati Egyetem, Szlovákia) és Dr. Turcsányiné Szabó Márta (ELTE, Budapest) a disszertációs munka elkészítéséhez nyújtott segítségüket, hasznos tanácsaikat és javaslataikat.

Absztrakt

Az algoritmusok tervezése és elemzése az első éves egyetemi hallgatók számára bonyolult feladat. Az animációk és vizualizációk segítségével való oktatás segíthet az elemi algoritmusok elsajátításában annak ellenére, hogy az eddig elvégzett kísérletek ezt nem minden esetben támasztják alá.

A doktori értekezés elméleti része összegzi az elektronikus tananyagba beágyazható, oktatási jellegű interaktív algoritmus-animációk és vizualizációk elkészítéséhez felhasználható elveket és javaslatokat.

A munka gyakorlati részében röviden foglalkozunk a programozás-oktatás lehetőségeivel virtuális világokban. Továbbá, bemutatunk egy saját fejlesztésű, algoritmus-animációkat és vizualizációkat tartalmazó oktatási portált. Végezetül, jellemezzük az általunk készített interaktív algoritmus-animációkat, majd tárgyaljuk a segítségükkel elvégzett pedagógiai kísérletek eredményeit.

Kulcsszavak: algoritmusok oktatása, interaktív animációk, algoritmusok animációi, programozás tanítása és tanulása

Abstract

Designing and analyzing algorithms is a hard task for undergraduate first-year students. Using animations and visualizations for understanding elementary algorithms might be helpful in education. However, results of pedagogical experiments in this field showed mixed results.

In the theoretical part of this doctoral dissertation, we summarize those principles and recommendations, which can be used to develop educationally effective algorithm animations. We focus mainly on the animations which can be embedded into electronic textbooks.

In the practical part, first, we briefly examine the possibilities of using virtual worlds for teaching programming. Next, we introduce an educational portal where we have collected algorithm animations and visualizations. Finally, we describe interactive algorithm animations developed by us, and we discuss the results of the pedagogical experiments which were conducted using these animations.

Keywords: teaching algorithms, interactive animations, algorithm animations, teaching and learning programming

Tartalomjegyzék

1 Bevezetés

A programozás az informatikaoktatás egyik fontos része. A számítástechnika szakos elsős egyetemi hallgatók számára azonban a programozás és algoritmikus gondolkodás elsajátítása az egyik legnehezebb feladat.

A programozás elsajátításához az alábbi négy fajta tudás szükséges (Bellstrom & Thoren, 2009):

- **Alapvető matematikai ismeretek** – ez lényegében az alap- és középiskolai matematika tananyagában található.
- **A programozói környezet (IDE) ismerete** – ez magában foglalja a programkód írásának, szerkesztésének, fordításának, hibakeresésének és futtatásának lépéseit.
- **A programozói nyelv ismerete** – ez a programozói nyelv (Pascal, C, C++, Java, PHP, JavaScript, stb.) utasításainak és szintaxisának ismeretét jelenti.
- **Az ismeret áttranszformálása a programozói logikába** – ez a problémamegoldást és a program logikájának kialakítását jelenti különböző utasítások és vezérlési szerkezetek kombinálásával. Ez lényegében különböző algoritmusok megértését, azok felhasználást, saját algoritmus kialakítását foglalja magában. Ide sorolhatjuk a diákok által megírt programban előforduló szemantikai és logikai hibák megkeresését és kijavítását is.

Ezek közül az első három elsajátítása a diákok számára többnyire nem okoz problémát, a negyedik, azonban sokszor nehéz, néha leküzdhetetlennek tűnő akadály. Ebben sokat segíthet, ha megismerik és megértik a programozási tételeket. Az elemi algoritmusok, tömbön végzett műveletek, rendezési algoritmusok megértésével fejlődik a kezdő programozók algoritmikus gondolkodása, hiszen ezen algoritmusok egyfajta sablonként és építőelemként is szolgálnak terjedelmesebb problémák megoldásához. Továbbá, a programozási tételek segítségével az egyes vezérlési szerkezetek gyakorlati alkalmazását is megtapasztalhatják a hallgatók.

Ezen doktori értekezés a felsorolt négy ismeret közül az algoritmusok megértésére, tehát a negyedik fajta ismeret elsajátításának megkönnyítésére fókuszál.

1.1 A témaválasztás indoklása

Az algoritmusok tervezése és elemzése a diákok számára bonyolult feladat. Ennek egyik fő oka, hogy az algoritmusok olyan folyamatokat írnak le, amely absztraktak és dinamikusak,

azonban a módszerek, melyekkel az algoritmusok oktatva vannak nem azok. Az animációk segítségével való oktatás segíthet ezen, hiszen követik az alábbi helyesnek tartott feltételezéseket:

- a grafikus szemléltetés jobb, mint a szöveges magyarázat;
- dinamikus grafika jobb, mint a statikus.

Azonban az eddig elvégzett kísérletek ezt nem minden esetben támasztják alá (Hansen, Narayanan, & Hegarty, 2002; C. Hundhausen & Douglas, 2000; C. D. Hundhausen, Douglas, & Stasko, 2002).

Young a kutatásai során foglalkozott a diákok által tanulással eltöltött, tanítási órákon kívüli idővel is. Publikációjában leírta, hogy a mai diákok annyira hozzászoktak a figyelmük elterelséhez (pl. médiákban megjelenő reklámokkal, képekkel, videókkal), hogy a múltbeli diákokhoz képest nehezebben tudnak koncentrálni (Young, 2002). A vizualizációkkal támogatott oktatás egyik előnye az is lehet, hogy segítségükkel a diákok jobban oda tudnak figyelni az animációkban szemléltetett folyamatokra (Grissom, McNally, & Naps, 2003).

A gondosan megtervezett animációk és vizualizációk növelik a hallgatók motiváltságát, a tanulást élvezetesebbé teszik, segítik a diákokat az új, rövid és hosszú távú ismeretek megszerzésében, továbbá csökkentik a hallgatók lemorzsolódását az egyetemeken (Urquiza-Fuentes & Velazquez-Iturbide, 2013).

Annak ellenére, hogy az ilyen algoritmus-animációk és vizualizációk segítik a tananyag megértését, az oktatók többsége mégsem használja őket. A tanárok többsége az alábbi okokat sorolta fel, amiért nem használnak animációkat az oktatásban (C. D. Hundhausen et al., 2002):

- Úgy érzik, nincs idejük utánanézni és megtanulni a használatukat.
- Úgy érzik, hogy nincs idő foglalkozni velük az órán, mivel ez más tanórai tevékenységtől venné el az időt.
- Úgy érzik, hogy animáció előkészítése a tanórákra túl sok erőfeszítésbe és időbe telik.
- Úgy érzik, hogy a használatuk nem hatékony az oktatásban.

A doktori értekezéssel kapcsolatos egyik célkitűzésünk egy olyan portál létrehozása volt, amely segítségével megkíséreltük az oktatásban sikeresen felhasználható algoritmus-animációk összegyűjtését és kategorizálását. Bízunk benne, hogy ezzel megkönnyítjük nem csak a diákok,

de az oktatók munkáját is, hiszen így egy helyen, kategóriákba sorolva érhető el több különböző algoritmus-animáció és vizualizáció. A tanárok a portál segítségével könnyen kereshetnek oktatásban hatékonyan felhasználható interaktív animációkat, kevesebb időt kell tölteniük a megfelelő animáció tanítási órába való beiktatásának előkészítésével.

1.2 Előzmények és célkitűzések

A több éves programozás oktatás során szerzett saját tapasztalataink is azt mutatják, hogy a diákoknak a legnagyobb gondot az algoritmusok megértése és az algoritmikus gondolkodás elsajátítása és fejlesztése okozza. Az algoritmusok megértésének megkönnyítésére próbáltunk különböző vizualizációkat és animációkat alkalmazni az oktatásban, köztük néhány saját fejlesztést is. Az e téren szerzett előzetes tapasztalatainkat és eredményeinket folyamatosan publikáltuk (Stoffa & Végh, 2006a, 2006b, 2014; Stoffová & Végh, 2007, 2010, 2014; Végh, 2006a, 2006b, 2010a, 2010b).

A doktori értekezés és a hozzá kapcsolódó kutatás célja, hogy összegezze az elektronikus tananyagba beágyazható, interaktív algoritmus-animációk tanulásra való hatását, ezek elkészítéséhez javasolt elveket, majd felmérje az animációk oktatásban való felhasználhatóságát a kezdő, programozást tanuló diákok (elsős informatika szakos hallgatók) körében. Konkrét céljaink:

- Azon elvek és módszertani javaslatok összefoglalása különböző irodalomforrásokból, amelyek alkalmazásával létrehozhatók jó minőségű, oktatásban sikeresen felhasználható interaktív animációk.
- A programozás-oktatás lehetőségeinek felmérése virtuális világokban, oktatásra felhasználható virtuális tér kialakítása Second Life környezetben, néhány interaktív animációs modell létrehozása ebben a környezetben, ezek használhatóságának felmérése.
- Egy saját, többnyelvű internetes portál létrehozása, majd annak feltöltése olyan algoritmus-animációkkal és vizualizációkkal, melyek beágyazhatók elektronikus tananyagba és megfelelnek az irodalomforrásokból összefoglalt elvek többségének. A portál elképzeléseink szerint hasznos kiindulópont lehet úgy a hallgatóknak, mint az oktatóknak. A diákok böngészhetnek az összegyűjtött animációk között, míg a tanárok azokat beágyazhatják saját elektronikus tananyagaikba, weboldalaikba.

- Olyan játék alapú interaktív animációk létrehozása, melyek segítségével a diákok hétköznapi objektumok (pl. kártyalapok, dobozok) segítségével megérthetik némely algoritmusok jellemző vonásait és az azok közti különbségeket, azonban a részletekbe nem mennek bele. A diákok a feladat megoldására játékos módon hozhatják létre a megoldás menetét (algoritmust), ezzel fejlesztve az algoritmikus gondolkodásukat.
- Mikro-szintű interaktív animációk létrehozását elősegítő JavaScript könyvtár fejlesztése, majd segítségével az elemi algoritmusokat, tömbműveleteket és rendezési algoritmusokat részletesen bemutató animációk létrehozása. Az ilyen interaktív animációk segítségével a diákok részleteiben, az algoritmusok forráskódjaival vagy pszeudokódjaival együtt ismerhetik meg a bemutatott algoritmusokat.
- A munka további célja felmérni a hallgatók véleményét és viszonyulását az ilyen interaktív animációk használatához az algoritmusok és a programozás oktatásánál.

A hallgatók véleményére vonatkozó hipotézis:

- **Hipotézis 1**: A megkérdezett informatika szakos hallgatók többsége szívesebben tanulna informatikai algoritmusokat animációk segítségével, mint statikus ábrák vagy szöveges magyarázat segítségével.

Az általunk elkészített „kártyák rendezése” (http://anim.ide.sk/kartyarendezes.php) interaktív animációkkal kapcsolatos hipotézis:

- **Hipotézis 2**: A „kártyák rendezése” interaktív animációk segítségével a diákok képesek felismerni az egyszerű rendezési algoritmusok lényeges lépéseit és az azok közti különbségeket.

Az általunk elkészített „ládák rendezése” (http://anim.ide.sk/ladakrendezese.php) interaktív animációval kapcsolatos hipotézis:

- **Hipotézis 3**: A „ládák rendezése” interaktív animációval a hallgatók képesek olyan rendezési algoritmust megalkotni, amelyben az eredetileg általuk megalkotott rendezési algoritmushoz képest lecsökken az összehasonlítások száma.

A rendezési algoritmusokat mikro-szinten szemléltető interaktív animációikkal (http://ani.ide.sk/) kapcsolatos hipotézis:

- **Hipotézis 4**: Az általunk elkészített interaktív animációkat használó hallgatók jobb eredményeket érnek el a teszteken, mint a statikus grafikus szemléltetést használó társaik.

2 A témával kapcsolatos eddigi kutatási eredmények áttekintése

Az animációk segítségével támogatott oktatással-tanulással több kutatás is foglalkozott az elmúlt években. Ezek alapján megállapítható, hogy az animációk használata nem minden esetben jár pozitív eredményekkel annak ellenére, hogy a diákok jobban kedvelik az animációk segítségével való ismeretelsajátítást, mint az animációk nélkülit (Byrne, Catrambone, & Stasko, 1999; Hansen et al., 2002; Kann, Lindeman, & Heller, 1997; Kehoe, Stasko, & Taylor, 2001).

Fontos, hogy az animációk megfelelő elemekkel szemléltessék az absztrakt fogalmakat. Megfelelő ábrázolás segítségével az animációk egyfajta hidat képezhetnek a valós világbeli tárgyak, szituációk és a programozással kapcsolatos fogalmak, folyamatok között (Rudder, Bernard, & Mohammed, 2007). Továbbá az is lényeges, hogy a hallgatók ne csupán passzív megfigyelői, hanem aktív résztvevői legyenek a vizualizációs folyamatoknak (Grissom et al., 2003; Katai & Tóth, 2010; T. L. Naps et al., 2002).

A témával kapcsolatos eddigi kutatások egy része kész vizualizációs rendszert használt az oktatás során (pl. JHAVÉ, Jeliot, BlueJ, ALVIS Live!, Balsa-II, Polka). Ezen rendszerek többsége egy vizuális elemekkel kiegészített hibakereső alkalmazásra hasonlít (Fleischer & Kucera, 2002). Az ilyen eszközök a beírt programkódban található adatokat és az azokon végzett műveleteket szemléltetik valamilyen standard formában (Lattu, Meisalo, & Tarhio, 2003). Előnyük, hogy a diákok a saját maguk által beírt, vagy módosított programkódot szemléltethetik grafikus formában, tehát szinte bármilyen programkód vizualizálható a segítségükkel. Hátrányuk, hogy az adatok és a folyamatok hasonlóan vannak szemléltetve minden esetben, nincsenek figyelembe véve az egyes algoritmusok jellegzetességei a folyamat vizuális megjelenítésekor.

A kutatások másik része olyan animációkat használt az oktatásban, melyek külön, egy-egy konkrét algoritmus bemutatására készültek. Bár ezek megtervezése és elkészítése jóval több időt vesz igénybe, de cserébe az animációk jobban kiemelhetik az egyes algoritmusok jellemzőit, fókuszálhatnak a fontosabb, jellegzetes lépésekre, több fajta interaktivitást tartalmazhatnak. Az ilyen, gondosan megtervezett, egy-egy adott algoritmusra fókuszáló animáció segítségével főleg a gyengébb képességű diákok értik meg könnyebben a vizualizált folyamatokat (Fleischer & Kucera, 2002; Kann et al., 1997). E doktori értekezésben is az ilyen fajta animációk használatára és létrehozására összpontosítottunk, szem előtt tartva az interaktivitás fontosságát.

2.1 A számítógépes oktatóanyagok fejlesztésének alapelvei

A hagyományos oktatással összehasonlítva, a számítógéppel támogatott oktatásnak több előnye is van. Ezek közé tartozik mindenekelőtt a tanulási folyamat individualizációja, beleértve a tananyagátadás sebességének személyre szabását. A számítógépes oktatóanyagok használata lehetővé teszi az interaktivitást, a tanári tevékenység megtöbbszörözését és automatizált eszközök használatát az oktatásban (Stoffová, 2004).

A számítógéppel támogatott oktatásnak vannak azonban hátrányai is. Ezek javarészt abból erednek, hogy szinte lehetetlen az oktatási folyamat egy univerzális modelljét megalkotni. Mivel a tanulási folyamat minden egyes lépésében nagyon nehéz megjósolni a diák minden lehetséges reakcióját, ezért az azokhoz megfelelő tanári válaszok modellezése is nagyon bonyolult, sokszor lehetetlen feladat (Stoffová, 2004).

B. F. Skinner, amerikai pszichológus 1954-ben fogalmazta meg a programozott tanulás fő elveit. Ebben az időben a programozott oktatás tankönyvek segítségével valósult meg. Skinner által megfogalmazott elvekből kiindulva, azok megfelelő módosításával és kiegészítésével eljuthatunk a számítógéppel támogatott oktatás alapelveihez, melyeket az alábbi táblázatban foglaltunk össze (Nádasi; Stoffová, 2004, 2005b):

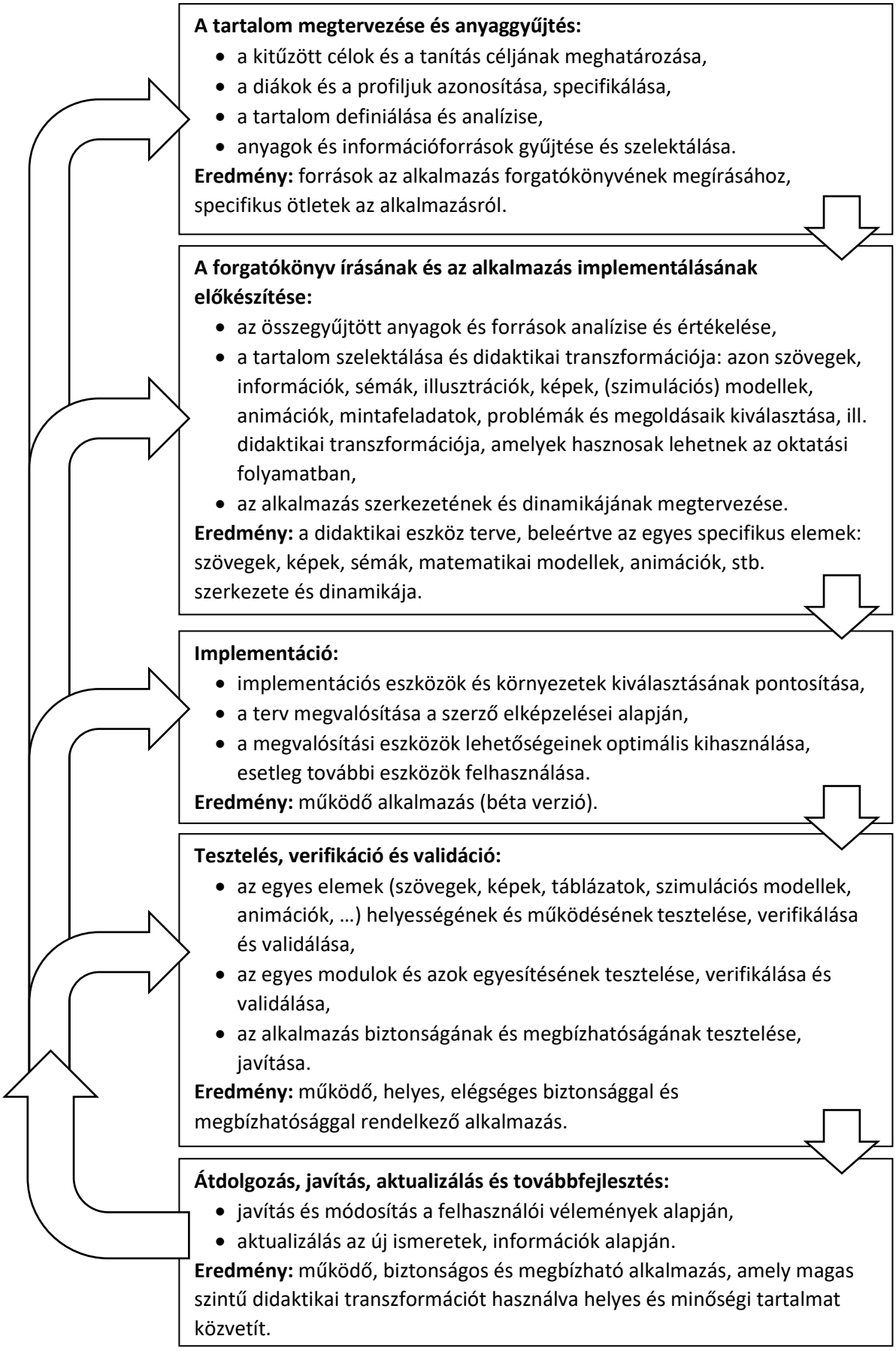
	Hagyományos programozott oktatási szituációk jellemzői (Skinner által megfogalmazott elvek)	**Számítógéppel támogatott oktatás jellemzői (a programozott oktatás módosított, kibővített alapelvei)**
1.	a diák kis lépésekben halad előre	a diák a számára megfelelő lépésekben halad előre
2.	a diáknak minden lépésben válaszolnia kell (nyílt, írásos válaszok)	interaktív, felfedezéses és aktív tanulás (gondolatban megfogalmazott válaszok)
3.	a diáknak a válaszai helyességéről azonnal információt kell szereznie	a diák teljesítményének folyamatos értékelése
4.	a diák saját ütemében halad a program feldolgozásában	a diák saját ütemében és saját tanulási módját alkalmazva halad előre
5.	a lépések gondosan meg vannak tervezve (lineáris vagy elágazásos stílusú programok)	a tanulási folyamat irányítása adaptív (a lineáris és elágazásos stílusú programok vegyes, adaptív használata)
6.	a diák önállóan dolgozza fel a programot	a tanítási folyamat optimalizálása a diák folyamatos értékelése alapján

7.	(verbális ingerek alkalmazása)	több érzékszerv bevonása a tanulási folyamatba, multimediális tananyag, hipertext szerkezet, animációk, stb. használata (verbális, vizuális és auditív ingerek alkalmazása)
8.	–	a tananyag tartalmának és a tanítás módjának folyamatos frissítése
9.	–	az implementáció átdolgozása, folyamatos jobbítása kihasználva az új rendelkezésre álló eszközöket, környezeteket és ismereteket a didaktikai alkalmazás intelligenciájának és minőségének növelésére

1. táblázat: A hagyományos programozott oktatás és a számítógépes oktatás jellemzőinek összehasonlítása

Jó minőségű elektronikus oktatóanyag elkészítése megköveteli a tanár (fejlesztő) különböző területeken szerzett jártasságait, készségeit és tapasztalatait. Egy didaktikai alkalmazás elkészítéséhez a pedagógiai és pszichológiai ismereteken kívül szükségesek programozói, tervezői, és grafikai ismeretek is (Stoffová, 2005a; Stoffová & Czakóová, 2016).

Mint minden szoftver, egy didaktikai alkalmazás elkészítése is egy életcikluson megy keresztül, amely az ötlet megszületésétől az alkalmazás fejlesztéséig, teszteléséig és aktív használatáig tart. Az egész szoftverfejlesztési folyamat a következő fázisokra osztható fel: megvalósíthatósági elemzés, követelményfeltárás (specifikáció), grafikai tervezés, konceptuális tervezés, kódolás (implementáció), tesztelés (verifikáció és validáció), működtetés és karbantartás. Sok esetben azonban a didaktikai szoftver létrehozása spirális fejlesztési modell alapján megy végbe, amely a tesztelés és a felhasználóktól (diákoktól) kapott visszajelzések alapján egyes fázisok megismétlését teszi lehetővé (Sommerville, 2007; Stoffová, 2005b; Stoffová & Czakóová, 2016). Oktatószoftverek készítésének általános iteratív modelljét szemlélteti az 1. ábra, amely a szoftverfejlesztési folyamat öt fázisát különbözteti meg (Stoffová, 2005a).



1. ábra: Didaktikai alkalmazás készítésének iteratív modellje (Stoffová, 2005a)

Egy oktatószoftver létrehozásának több oka is lehet, például:

- a tanagyag és a benne található összefüggések könnyebb elképzelése és megértése,
- a tananyag hatékony megerősítése számítógépes alkalmazás segítségével,
- a diák motiválása a probléma kreatív megoldására,
- kellemesebb légkör létrehozása a tanítási órákon, a diákok feszengésének feloldása és aktivitásuk növelése,
- individuális tanulás a diák saját ütemében,
- folyamatos visszajelzés a tanulás sikerességéről,
- konstruktív tanulás didaktikai játékok, szimulációk segítségével, stb. (Stoffa, 2003; Stoffová & Czakóová, 2016)

Az alkalmazáskészítés első fázisában a tanárnak pontosítania kell az elképzeléseit a tananyag tartalmáról és mértékéről, meghatároznia azon célokat, melyeket az alkalmazás segítségével el szeretne érni a tanítás során. Fontos azt is meghatározni, hogy az oktatószoftver a tanítási-tanulási folyamat melyik fázisára fog összpontosítani (pl. magyarázat, prezentálás, szemléltetés, interaktív gyakorlás, tesztelés, új ismeretek szerzése szimulációk segítségével, vagy csupán a tanítási óra élvezetesebbé varázslása).

A követelmények analízise és specifikációja alapján szükséges meghatározni az alkalmazás és a felhasználó közötti kommunikációs felületet és a felhasználótól elvárt tevékenységeket, beleértve az interakció módját is. A tervezési fázisban egyrészt az alkalmazást grafikailag is meg kell tervezni (egyes ablakok megtervezése, objektumok elrendezése a képernyőn, animációk fázisainak megtervezése), másrészt az alkalmazással végezhető műveletekre, tevékenységekre is oda kell figyelni (szimulációk, interaktivitás megtervezése).

A konceptuális tervezés során a fejlesztő átgondolja, hogyan lehet leprogramozni az egész alkalmazást a kiválasztott programozási nyelven.

Az implementálás során megírt forráskód tiszta, áttekinthető legyen. A későbbiekben a tananyag frissítésekor, ill. a diákoktól kapott visszajelzések alapján gyakran előfordulhat, hogy a programkódban módosításokat szükséges végrehajtani.

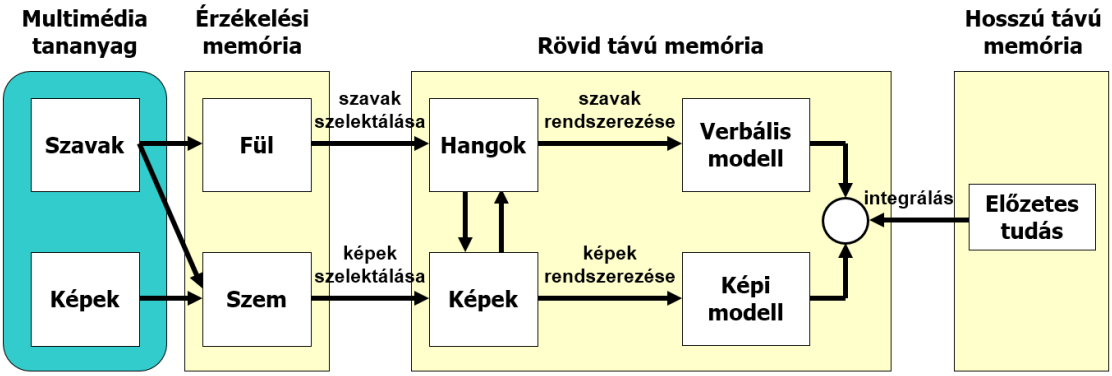
Fontos a tesztelésre és az alkalmazás finomítására, aktualizálására is elegendő időt szánni. A tesztelés során a tanár figyeli, hogy a diákokat mennyire érdekli az alkalmazás, szívesen dolgoznak-e vele, ill. segít-e a meghatározott oktatási célok elérésében (Stoffová & Czakóová, 2016).

Maga a didaktikus szoftverfejlesztés egy valódi iterációs folyamat, úgy, ahogy az 1. ábrán a nyilak jelzik. Bármely lépésben szükség szerint megismételhetjük, bővíthetjük, átdolgozhatjuk az előbbi lépések tartalmát. Az implementáció átdolgozása, folyamatos jobbítása kihasználva az új rendelkezésre álló eszközöket, környezeteket és ismereteket a didaktikai alkalmazás intelligenciájának és minőségének növelését eredményezheti.

Maga a fejlesztő dönti el meddig érdemes és eredményes egy bizonyos koncepció alapján jobbítani és bővíteni az alkalmazást és mikor szükséges magát a koncepciót is megváltoztatni.

2.2 A multimédia tanulás kognitív elmélete és Mayer multimédia elvei

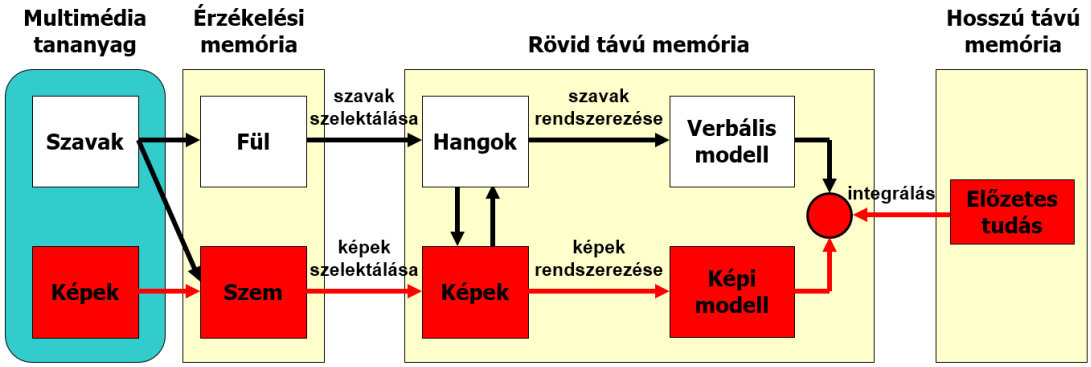
Mayer a kutatásai alapján kidolgozta a multimédia tanulás kognitív elméletét (Cognitive Theory of Multimedia Learning), amelyet az alábbi ábra szemléltet. Ezen információk hasznosak lehetnek bármilyen elektronikus tananyag, oktatószoftver, animáció vagy szimuláció megtervezéséhez. Az alábbi ábrán jól látható, hogy az emberi agy hogyan dolgozza fel az információkat.



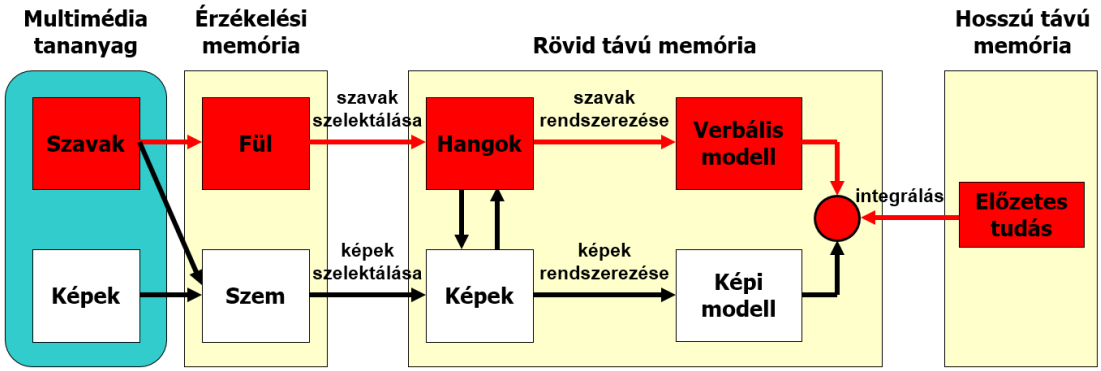
2. ábra: A multimédia tanulás kognitív elmélete (Mayer, 2009)

A különböző formában megjelenő információ különbözőképpen van feldolgozva. Az algoritmus-animációk elsősorban a vizuális csatornát használják az információ továbbítására (Esponda-Arguero, 2010). Az animációk azonban ki lehetnek egészítve magyarázattal nyomatott szöveg vagy beszéd formájában, így az információ feldolgozása audio- és vizuális csatornán is végbe mehet.

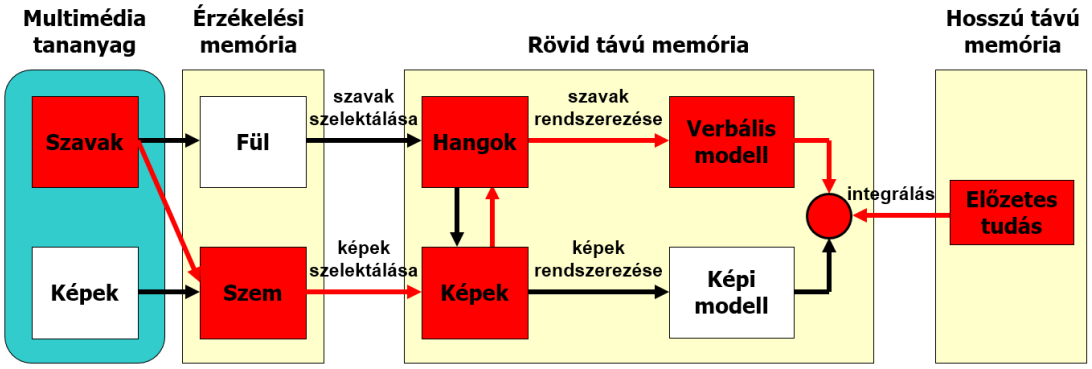
A következő ábrák a kép, beszéd és a nyomtatott szöveg feldolgozásának folyamatát szemléltetik.



3. ábra: Kép feldolgozásának folyamata



4. ábra: Beszéd feldolgozásának folyamata



5. ábra: Nyomtatott szöveg feldolgozásának folyamata

A multimédiás tananyagok kialakítása során a cél, hogy az információ különböző csatornákon legyen feldolgozva, tehát a legjobb megoldásnak a kép és a beszéd egyidejű használata tűnik. Sok esetben azonban a beszédet érdemesebb nyomtatott szöveggel helyettesíteni (pl. ha az adott nyelv nem a diákok anyanyelve; a szöveg olyan információt tartalmaz, melyet a diákoknak szükséges mélyebben átgondolniuk, szükség esetén bármikor újra elolvasniuk). Ilyenkor azonban figyelni kell arra, hogy elegendő idő legyen a nyomtatott

szöveg és a képi információ feldolgozására is, hiszen a folyamat nem két teljesen párhuzamos csatornán megy végbe (3. és 5. ábra).

Mayer a kutatásai alapján 12 elvet fogalmazott meg, melyeket érdemes figyelembe venni a multimediális tananyagok elkészítésekor. Ezeket az elveket három fő kategóriába csoportosította (Mayer, 2009):

I. A tárgyhoz nem tartozó információk feldolgozásának csökkentése:

    1. Összefüggés elve (Coherence Principle)
    2. Kiemelés elve (Signaling Principle)
    3. Redundancia elve (Redundancy Principle)
    4. Térbeli összefüggés elve (Spatial Contiguity Principle)
    5. Időbeli összefüggés elve (Temporal Contiguity Principle)

II. A lényeges információk feldolgozásának irányítása:

    6. Szegmentáció elve (Segmenting Principle)
    7. Előképzés elve (Pre-training Principle)
    8. Módbeliség elve (Modality Principle)

III. Információk konstruktív feldolgozásának elősegítése:

    9. Multimédia elve (Multimedia Principle)
    10. Társalgási stílus elve (Personalization Principle)
    11. A magyarázat hangjának elve (Voice Principle)
    12. Az oktató megjelenítésének elve (Image Principle)

Az alábbiakban ezen elveket mutatjuk be röviden, melyek részletes tárgyalása megtalálható a (Mayer, 2009) könyvben. Fontos megjegyeznünk, hogy ezen elvek használata nem feltétlenül jár pozitív eredményekkel az oktatási-tanulási folyamatban. Mayer mindegyik elv tárgyalásánál részletesen rámutat arra, milyen feltételek teljesülése mellett járhat pozitív, semleges vagy negatív eredményekkel az adott kísérlet (tanulási folyamat).

1. Összefüggés elve (Coherence Principle)

A diákok hatékonyabban tanulnak, ha a tárgyhoz nem tartozó (külső) információk nincsenek a tananyagban. A tanulás hatékonyabb, ha:

- az érdekes, de irreleváns szavak és képek nincsenek a multimédia tananyagban;

- az érdekes, de irreleváns hangok és zene nincs a multimédia tananyagban;
- a felesleges (pl. részletes) szöveg és a jelek nincsenek a tananyagban.

2. **Kiemelés elve (Signaling Principle)**

   A diákok hatékonyabban tanulnak, ha a fontos információk a tananyagban valamilyen formában ki vannak emelve.

3. **Redundancia elve (Redundancy Principle)**

   A diákok hatékonyabban tanulnak illusztrációból és szóbeli magyarázatból, mint illusztrációból, szóbeli magyarázatból és írott szövegből. Szóbeli magyarázat és írott szöveg egyidejű megjelenítése felesleges, sőt negatív hatásokkal járhat a tanulási folyamatban.

4. **Térbeli összefüggés elve (Spatial Contiguity Principle)**

   A diákok hatékonyabban tanulnak, ha a logikailag egymáshoz közel tartozó szavak és képek a képernyőn (könyvben) is minél közelebb vannak egymáshoz.

5. **Időbeli összefüggés elve (Temporal Contiguity Principle)**

   A diákok hatékonyabban tanulnak, ha a szavak és a képek nem egymás után, hanem egyidejűleg vannak prezentálva.

6. **Szegmentáció elve (Segmenting Principle)**

   A diákok hatékonyabban tanulnak, ha a tananyag kisebb logikai részekre van bontva, melyeket a felhasználó egymás után, saját tempóval nézhet meg.

7. **Előképzés elve (Pre-training Principle)**

   A diákok hatékonyabban tanulnak, ha már ismerik a fő fogalmakat és azok jellemzőit.

8. **Módbeliség elve (Modality Principle)**

   A diákok hatékonyabban tanulnak képekből és elmondott szövegből, mint képekből és nyomtatott szövegből.

9. **Multimédia elve (Multimedia Principle)**

   A diákok könnyebben tanulnak szavakból és képekből, mint csupán szavakból.

10. Társalgási stílus elve (Personalization Principle)

A diákok könnyebben megértik a multimédia tananyagot, ha a benne szereplő magyarázat társalgási, személyes stílusban található formális stílus helyett.

11. A magyarázat hangjának elve (Voice Principle)

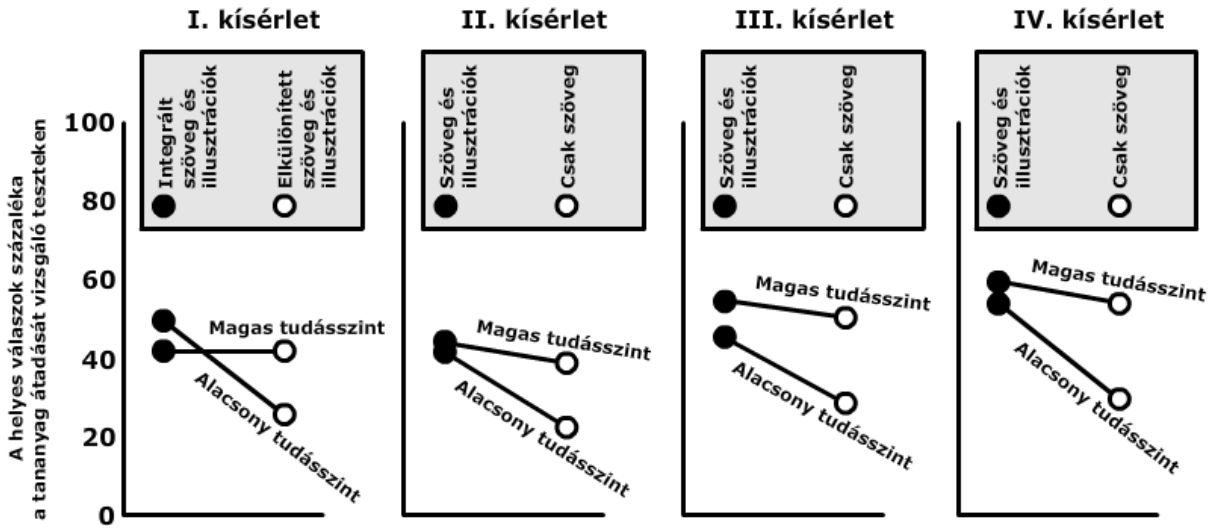
A diákok hatékonyabban tanulnak, ha a magyarázatot gépi (szintetizált) hang helyett kellemes emberi hangon hallják.

12. Az oktató megjelenítésének elve (Image Principle)

A tanulás nem feltétlenül hatékonyabb, ha a magyarázó személy képe meg van jelenítve a képernyőn.

A felsorolt elvek alkalmazásakor figyelembe kell vennünk többek között a tananyag típusát és a diákok tudásszintjét, nyelvtudását, tanulási stílusát is a hatékonyabb tanulás eléréséhez. Pl. gyengébb diákoknál jobb eredmények érhetők el az egyes elvek alkalmazásával, míg a magasabb tudásszinttel rendelkező diákoknál nem figyelhető meg lényeges javulás (Mayer, 2009).

Az alábbi ábra 4 különálló kísérlet eredményét szemlélteti különböző tudásszinttel rendelkező diákoknál. A kísérletekben a diákok két fajta tananyaggal dolgoztak. Az egyik diákcsoportban a tananyag megfelelt a multimédia elveknek (szöveget és illusztrációt is tartalmazott), míg a másik diákcsoportban a tananyag gyengébb minőségű volt (a szöveg és az illusztráció elkülönített helyen szerepelt, ill. a tananyag csupán szöveget tartalmazott).



6. ábra: Magas és alacsony tudásszinttel rendelkező diákokon végzett kísérletek eredményei

A grafikonokból kiolvasható, hogy a kísérletekben vizsgált multimédia elvek használata főleg a gyengébb képességű, alacsonyabb tudásszinttel rendelkező diákoknál hozott lényeges javulást a tananyag megértésében (Mayer, 2009).

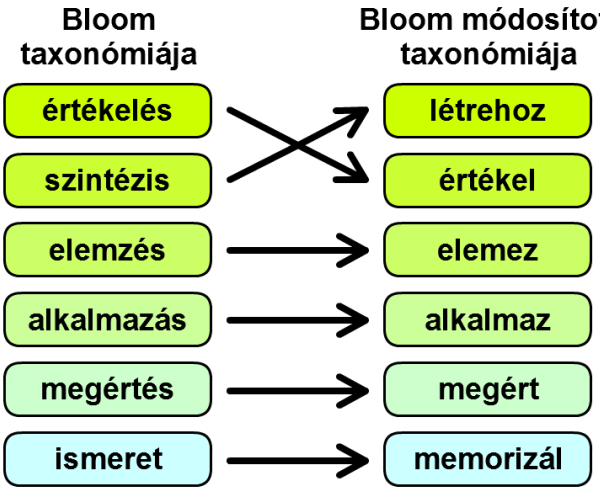
Egy másik kísérlet eredménye is azt bizonyította, hogy a magasabb évfolyamba járó diákok számára nem olyan fontos a multimédia tananyagban a kép megjelenítése, mint az első évfolyamos hallgatóknak. Ez azzal magyarázható, hogy a tapasztaltabb diákok könnyebben tudnak kialakítani a rövid távú memóriájukban egy belső képet a magyarázatokhoz, mint a kezdő hallgatók. Ezen kísérlet eredménye azonban azt is kimutatta, hogy azon hallgatók, akik látták az animációt, sokkal rövidebb idő alatt végeztek a utóteszt kitöltésével (Kehoe et al., 2001).

A módbeliség elve sem használható minden esetben sikeresen. Egy kísérlet során azon hallgatók, akik hanganyaggal kiegészített animációt láttak hasonlóan teljesítettek, mint azok, akik az animációt szöveges magyarázattal kiegészítve használták. Az egyik lehetséges oka annak, hogy a hanggal kiegészített animációt használók nem teljesítettek jobban az lehet, hogy ezek a diákok nem figyeltek annyira oda az elhangzó magyarázat részleteire, mint azok a diákok, akiknek ugyanezt el kellett olvasniuk (Kehoe et al., 2001; Palmiter & Elkerton, 1991). Ez alapján arra következtethetünk, hogy a módbeliség elvének használata olyan tananyagnál, ahol fontos odafigyelni a részletekre, nem feltétlenül vezet jobb eredményhez.

Bár az algoritmusok animációinak megtervezésekor fontosak a multimédia elvek betartása és az adatszerkezet megfelelő grafikus reprezentációja, az interaktivitás sokkal fontosabbnak tűnik. A következő fejezetekben tárgyalt kutatások is azt bizonyítják, hogy az animációk oktatásban való felhasználása során jóval lényegesebb a diákok aktív részvétele a vizualizációs folyamatban, mint a tananyag grafikus szemléltetésének a módja.

2.3 Bloom módosított taxonómiája a programozás tanítására és tanulására vonatkoztatva

Bloom eredeti taxonómiája (Bloom, Englehart, Furst, Hill, & Krathwohl, 1956) hat egymásra lineárisan épülő kognitív szintet tartalmaz, mely 45 év eltelte után módosításokon esett át (Anderson et al., 2001). Az eredeti és a módosított taxonómiát az alábbi ábra szemlélteti.



7. ábra: Bloom eredeti és módosított taxonómiája

Ezen taxonómiát alkalmazva pontosabban meghatározható, hogy a diákoktól a tanulási folyamatban milyen kognitív követelmények várhatók el. Az alábbi felsorolásban Bloom módosított taxonómiájának egyes szintjeihez megpróbáljuk hozzárendelni az algoritmusok és az adatszerkezetek tanulása során megfigyelhető kognitív aktivitások eredményeit (T. L. Naps et al., 2002):

1. szint: memorizálás – A diákok ismerik az adatszerkezeteket a nevük alapján (pl. tömb, gráf, fa, kupac) és az algoritmusok neveit (pl. beszúró rendezés, gyorsrendezés).

2. szint: megértés – Ismerik az algoritmusok működésének a lényegét, el tudják ábrákkal és saját szavaikkal magyarázni őket. Meg tudják írni az algoritmust ugyanabban a programozási nyelvben, amelyben tanulták, le tudják tesztelni, hogy a program működik-e. Megértik és el tudják ismételni az algoritmus legjobb és legrosszabb esetének az elemzését.

3. szint: alkalmazás – Tudják alkalmazni az előzőleg elsajátított algoritmusokat bizonyos esetekben, más programozási környezetekben, vagy speciális bemeneti adatokon. Egyszerű algoritmusokra el tudják készíteni a legjobb és a legrosszabb eset elemzését.

4. szint: elemezés – A diákok értik ugyanolyan, vagy hasonló problémákat megoldó algoritmusok között a kapcsolatokat. Képesek érvekkel alátámasztani és/vagy bizonyítani az algoritmusok helyességét. Képesek elemezni bonyolultabb problémákat, beazonosítani bennük lényeges objektumokat, majd a problémákat felbontani kisebb, kezelhető problémákra.

5. szint: értékelés – Meg tudják tárgyalni az előnyeit és a hátrányait olyan különböző algoritmusoknak, melyek ugyanolyan vagy hasonló problémákat oldanak meg. Át tudják gondolni, hogy miért és hogyan kellene valamilyen algoritmust módosítani vagy

összekombinálni más algoritmusokkal ahhoz, hogy hatékonyan megoldjanak egy új, bonyolultabb problémát.

6. szint: létrehozás – A diákok képesek olyan bonyolultabb, komplex problémák megoldására, amelynél többféle adatszerkezet, algoritmus és technika egyidejű használata szükséges.

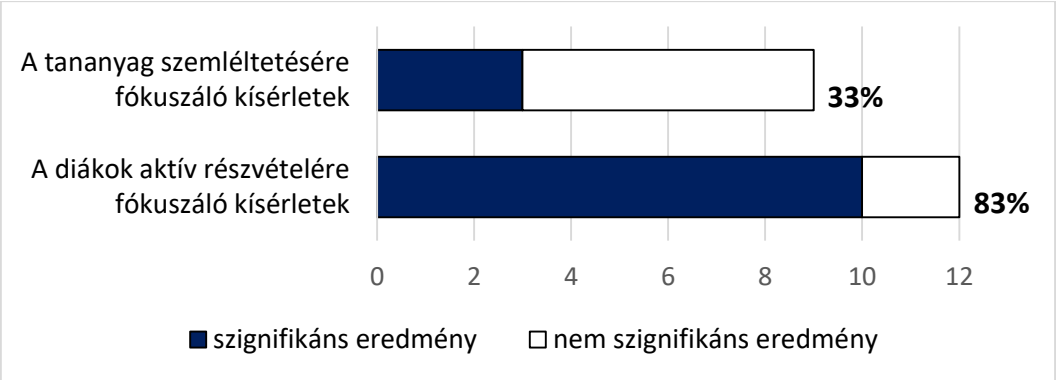
Ahogy Bloom is hangsúlyozta, fontos hogy a diákok fokozatosan, egymás után sajátítsák el az egyes szintekhez tartozó aktivitásokat. Az algoritmusok interaktív animációi főleg az alacsonyabb szinteken (1. – 2.) alkalmazhatók sikeresen az oktatási-tanulási folyamatban. Ennek ellenére, az animációk használata javasolt az oktatásban, hiszen ezek segítségével a diákok biztosabban és gyorsabban megértik az alapokat. A további szintek elérése ezek után könnyebb lesz számukra, így a magasabb szintű aktivitásokra több idő maradhat a tanórából (Grissom et al., 2003).

Kann és társai által végzett kísérlet (Kann et al., 1997) eredménye azt támasztja alá, hogy az algortimusok animációinak összekapcsolása az algoritmusok implementálásával, effektív módja lehet az algoritmusok elsajátításának. Azon diákok, akik a feladatlap mellett látták az animációt, majd implementálták az algoritmust Ada programozási nyelven, sokkal jobban teljesítettek az utótesztelésen, mint azok, aki az animáció megtekintése nélkül oldották meg a feladatot. Ez a kísérlet is arra utal, hogy az animációk használata hasznos lehet Bloom taxonómiájának első két szintjével kapcsolatos tanulási folyamatokban. A kísérlethez hasonlóan, az oktatás során is az animációk megtekintését az algoritmusok implementálása követheti, amely már Bloom taxonómiájának 3. szintjéhez tartozik.

2.4 Az interaktivitás fontossága az algoritmusok animációiban

Több különböző kutatás eredménye is azt sugallja, hogy a diákok számára csak akkor jelent számottevő segítséget a vizualizációk használata az algoritmusok megértésében, ha a diákok aktívan is részt vesznek benne, tehát nem csupán passzív megfigyelői az animációknak (Furcy, Naps, & Wentworth, 2008; Grissom et al., 2003; Kuk et al., 2012; T. L. Naps et al., 2002; Patwardhan & Murthy, 2015; Stoffa, 2003).

Az alábbi grafikon 21 kísérlet eredményét szemlélteti, melyek teljes meta-analízise megtalálható a (C. D. Hundhausen et al., 2002) publikációban. A vizualizációk hatékonyságát vizsgáló kísérletek közül 9 darab a szemléltetésre fókuszált, míg 12 darab az interaktivitást helyezte vizsgálat alá.



8. ábra: A vizualizációk hatékonyságát vizsgáló kísérletek eredményei az szerint kategorizálva, hogy a kísérlet szemléltetésre vagy interaktivitásra fókuszált-e

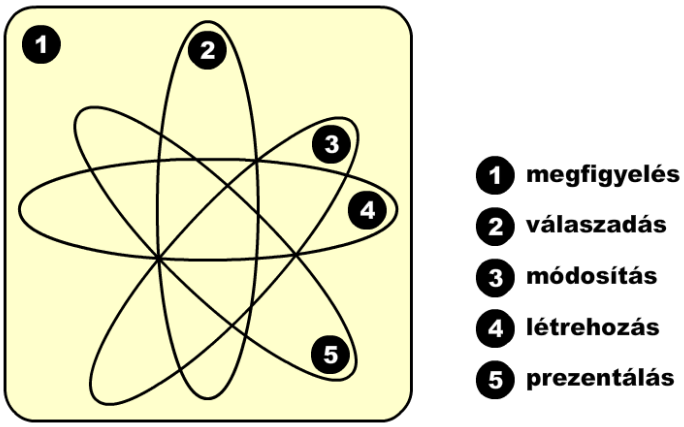
A grafikonból kiolvasható, hogy az animációkkal támogatott oktatás során nagyobb hatással van a tanulásra a diákok aktív részvétele, mint az, amit a diákok a képernyőn látnak – ez a megfigyelés összhangban van a konstruktív tanulás elméletével (C. Hundhausen & Douglas, 2000; C. D. Hundhausen et al., 2002; T. L. Naps et al., 2002). Az interaktivitás a vizualizált szimulációs modell használatánál aktív tanuláshoz vezet, a diák maga ellenőrizheti szimulációs kísérletek elvégzésével a probléma megértésének és megoldásának helyességét (Stoffová, 2004).

Az animációkban az interaktivitásnak különböző szintjei lehetnek: az egyszerű megfigyeléstől, az animáció egyes részeinek módosításain keresztül, egészen a saját animáció elkészítéséig. Az alacsonyabb szintű interaktivitással rendelkező animációk a behaviorista tanítási módra fókuszálnak, míg a magasabb szintű interaktivitással rendelkező animációk magasabb fogalmi és procedurális tanulást eredményeznek (Patwardhan & Murthy, 2015). Elsősorban az utóbbi tanulási formák vezetnek kognitív folyamatokhoz és aktív tanuláshoz (Urquiza-Fuentes & Velazquez-Iturbide, 2013). Ezen megfigyelések is azt hangsúlyozzák, hogy az interaktivitásnak lényeges szerepe van az animációkkal és a vizualizációkkal támogatott ismeretelsajátításban.

A hallgatók vizualizációs folyamatban való részvétele lehet különböző féle (Furcy et al., 2008; Grissom et al., 2003):

- megfigyelés,
- válaszadás,
- módosítás,
- létrehozás,
- prezentálás.

A kapcsolat ezek között nem lineáris, az alábbi Venn diagram szemlélteti e tevékenységek egymással való kapcsolatait. A „megfigyelés” kitölti az egész teret, hiszen ez mindegyik aktivitásnál jelen van valamilyen formában. A maradék négy tevékenység bármilyen kombinációban előfordulhat a vizualizációs folyamat során.



9. ábra: A vizualizációs folyamatban való részvétel típusainak lehetséges átfedései (T. L. Naps et al., 2002)

A **megfigyelés** önmagában a legpasszívabb aktivitás (Grissom et al., 2003). Ez az aktivitás azonban mindegyik tevékenységnél előfordul, sőt a legkülönbözőbb formákban lehet jelen. Pl. a diák lehet passzív megfigyelője az animációnak, de megfigyelheti az animációban történő folyamatok irányát, sebességét is; fókuszálhat az animált objektumokra, a programkódra, vagy a ciklusváltozók értékeire; esetleg foglalkozhat a magyarázó szöveg elolvasásával vagy meghallgatásával, majd annak megértésével (T. L. Naps et al., 2002).

A **válaszadás** során a diákok megválaszolhatják az animációval kapcsolatos kérdéseket (Furcy et al., 2008). Ezek általában olyan kulcsfontosságú kérdések, melyek szükségesek az adott animáció megértéséhez. Kérdések beiktatásával a vizualizációs folyamatba a diákokat ösztönözhetjük arra, hogy aktívabban figyelemmel kísérjék az animációt és mélyebben elgondolkodjanak a vizualizált algoritmuson. Ilyen kérdések lehetnek például (T. L. Naps et al., 2002):

- Mi lesz a következő lépés a vizualizációban? (előrejelzés)
- A forráskód melyik része tartozik az éppen vizualizált folyamathoz? (kódolás)
- Mi az vizualizáció által bemutatott algoritmus lefutásának legjobb és legrosszabb esete? (hatékonyság analizálása)
- A vizualizáció által bemutatott algoritmus hibamentes? (hibakeresés)

Bár a kérdezés lehet az animáció része, azonban ez nem feltétlenül szükséges. A kikérdezés megvalósulhat papír és ceruza használatával is (Grissom et al., 2003). Sok esetben az utóbbi még előnyösebb is lehet, mivel a diák saját maga döntheti el, hogy mikor fog válaszolni a kérdésekre, ill. a kérdések megválaszolása előtt többször is végignézheti saját tempójában az animációt, kísérletezhet az interaktív elemekkel.

Az sem minden esetben lényeges, hogy a diáktól kapunk-e szóbeli vagy írásbeli visszacsatolást. Bizonyos esetekben elegendő, ha a kérdés feltevésével elérjük, hogy a diák átgondolja az algoritmust (Hansen et al., 2002).

A **módosítás** alatt értjük például, ha a vizualizáció elején a diákok megadhatják a bemeneti adatokat. Ennek köszönhetően a diákok kísérletezhetnek a bemenettel és megvizsgálhatják, hogy különböző eseteken hogyan fut le az algoritmus (Furcy et al., 2008; T. L. Naps et al., 2002). A bemeneti adatok kezdeti megadásán túl még hatékonyabb lehet az animáció megértése, amennyiben a diákok menet közben is módosíthatják az adatokat, természetesen bizonyos feltételek mellett. Ügyelni kell arra, hogy az animáció alatt az algoritmus által már elvégzett műveletek eredményei ne legyenek módosíthatók, pl. egy rendezési algoritmus során a már rendezett rész elemeinek értékét ne lehessen megváltozatni.

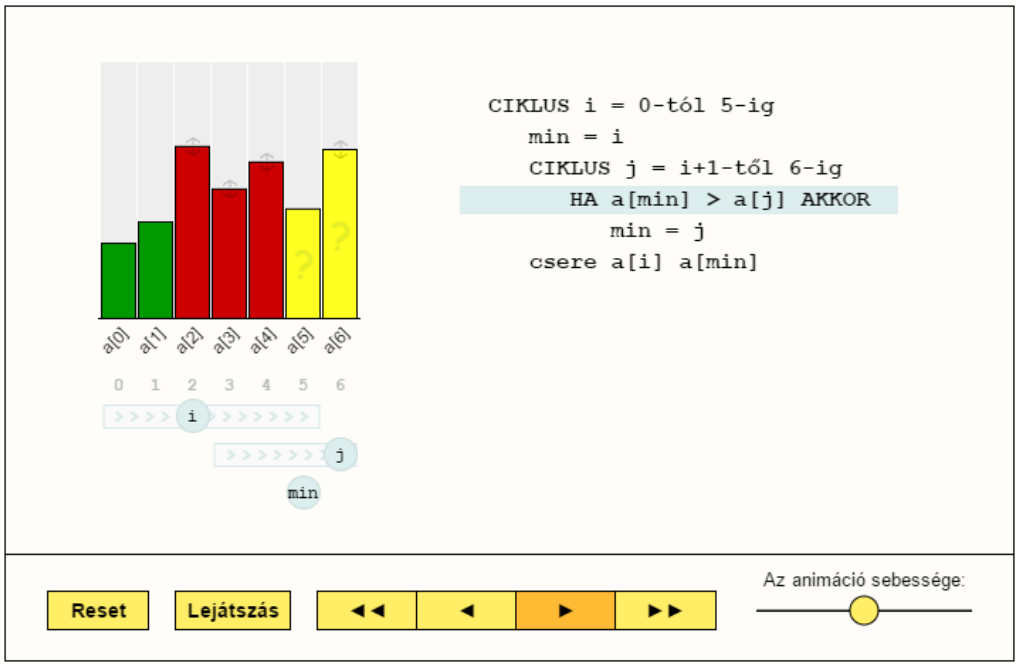
A **létrehozás** folyamán a diákok saját maguk készítik el valamilyen formában az adott algoritmussal kapcsolatos vizualizációt (Furcy et al., 2008; C. D. Hundhausen & Brown, 2008). Ezt elkészíthetik programkód megírásával valamilyen programozási nyelvben (pl. Java, JavaScript, Delphi, C++) vagy valamilyen vizualizációs rendszerben (pl. JHAVÉ, Jeliot, BlueJ, ALVIS Live!). Egy másik megoldás lehet valamilyen grafikus editor és animáció készítéséhez használható szoftver alkalmazása (pl. Adobe Flash, PowerPoint, Prezi). A létrehozás folyamatába tartozik azonban az is, ha a diákok egyszerű, mindennapi eszközök (pl. papír, olló, ceruza) segítségével készítik el az animációs modellt. Az ilyen módszer alkalmazása a diákot egy virtuális számítógép szerepébe teszi, hiszen neki kell a vizualizációt lefuttatnia az adott algoritmus alapján (C. Hundhausen & Douglas, 2000; T. L. Naps et al., 2002).

A **prezentálás** során a diák előbb bemutatja a vizualizációt nézőközönség előtt, majd annak megbeszélése következik (Furcy et al., 2008; C. D. Hundhausen & Brown, 2008). A bemutatott vizualizáció lehet a hallgató saját munkája, azonban ez nem feltétlenül szükséges (T. L. Naps et al., 2002).

A multimédia elvek megfelelő kombinációjának használatához hasonlóan, itt is fontos a megfelelően összekombinált interaktivitás-típusok kiválasztása egy-egy animációhoz. Ezek

közül túl sok fajta aktivitást igénylő cselekvések egyidejű alkalmazásával nem feltétlenül érhetünk el pozitív eredményt, mivel az a diákok figyelmének túlterheléséhez vezethet.

Vegyük például az alábbi ábrán látható, egy olyan rendezési algoritmust bemutató animációt, amely párhuzamosan több információt is megjelenít a képernyőn (az oszlopokkal szemléltetett tömböt, ciklusváltozókat, algoritmus pszeudokódját). Az animációban használt adatok módosíthatók (az oszlopok vertikális átméretezésével) és az animáció az alsó sávban található vezérlőgombok segítségével irányítható (reset, lejátszás, lépés hátra, előre, stb).



10. ábra: A minimumkiválasztásos rendezési algoritmust bemutató interaktív animáció

A felhasználó az animációt saját gondolkodási sebességének megfelelően próbálja meg értelmezni, közben összeköti az oszlopokkal szemléltetett tömböt a ciklusváltozókkal és a pszeudokóddal. Tapasztalataink alapján, amennyiben a feltételvizsgálat során még kvíz kérdéseket is beiktatnánk az animációba (pl. „Igaz vagy hamis a feltétel?”, „Módosítsd az a[j] értékét úgy, hogy a feltétel igaz legyen!”), az már a diákok számára kellemetlenné, idegesítővé válna. A folyamatosan megjelenő kvíz kérdések elterelnék a diákok figyelmét egyrészt az animációban egymás mellett párhuzamosan szemléltetett információk (tömb, ciklusváltozók, pszeudokód) összekapcsolásától, másrészt az algoritmus korábbi lépéseinek az aktuális és az elkövetkező lépéseivel való összefűzésétől.

Amennyiben a diákok elvesztik a gondolatmenetüket, legtöbbjük a további kérdésekre csupán tippeléssel válaszol, a kérdések alapos értelmezése nélkül (Grissom et al., 2003).

2.5 A programozás és az algoritmusok oktatásánál sikeresen felhasználható interaktív animációk jellemzőinek összefoglalása

Az interaktív animációk felhasználása az informatika oktatásában az 1970-80-as évek óta figyelhető meg (T. L. Naps et al., 2002; Urquiza-Fuentes & Velazquez-Iturbide, 2013). Az elmúlt 30-35 évben végzett kutatások során kialakultak különböző „bevált gyakorlatok". Az elkövetkező alfejezetekben összefoglaljuk azokat a jellemzőket, melyekkel egy oktatásra szánt algoritmus-animációnak érdemes rendelkeznie. Ezeket a javaslatokat két csoportba különítettük el. Az első csoport foglalkozik az animáció dizájnjával, annak külső megjelenítésével. A második csoport az interaktivitással foglalkozik, amely segítségével a diákokat aktív résztvevőkként lehet bevonni a vizualizált megoldás folyamatába.

Természetesen az, hogy az alábbi alfejezetekben felsorolt javaslatok közül melyeket érdemes megvalósítani egy adott animáció elkészítésénél, ill. az milyen formában legyen jelen az animációban, jelentős mértékben függ a szemléltetni kívánt tananyagtól és a diákok előzetes tudásszintjétől is. Mivel minden diák másképp gondolkodik és más tanulási stílussal rendelkezik, nem létezik olyan vizualizációs rendszer vagy algoritmus-animáció, amely a „legjobb" az összes diák számára (T. L. Naps et al., 2002).

Véleményünk szerint több különböző, gondosan megtervezett interaktív animáció segítségével azonban szinte az összes diáknál elérhető, hogy könnyebb és érdekesebb legyen számukra a tananyag elsajátítása. Egy algoritmushoz három különböző animáció használatának előnyére mutattak rá Hansen és társai is (Hansen et al., 2002), akik szerint célszerű az alábbi három típusú animáció egymás utáni alkalmazása az oktatásban:

1. Az első típusú animáció **az algoritmus működésének lényeges eseményeit** mutatja be, nem megy bele a részletekbe. Ezen animációk egyrészt megalapozzák az absztrakt elemek megértését valós világbeli példákon keresztül, másrészt motivációként is szolgálnak (Hansen et al., 2002). A valós világbeli példák használata a programozás oktatásánál elősegíti, hogy a diákok könnyebben megértsék és alkalmazzák az absztrakt fogalmakat. Ez által a diákok már a kezdetben, amikor még csak ismerkednek az algoritmussal, a valós világ és a logikai programozás közötti kapcsolat kialakítására vannak ösztönözve (Bernát, 2014; Rudder et al., 2007). Az általunk elkészített animációk közül ilyen a rendezési algoritmusok oktatásánál felhasználható, 7. és 8. fejezetekben tárgyalt, kártyák és ládák rendezésére szolgáló interaktív animációk.

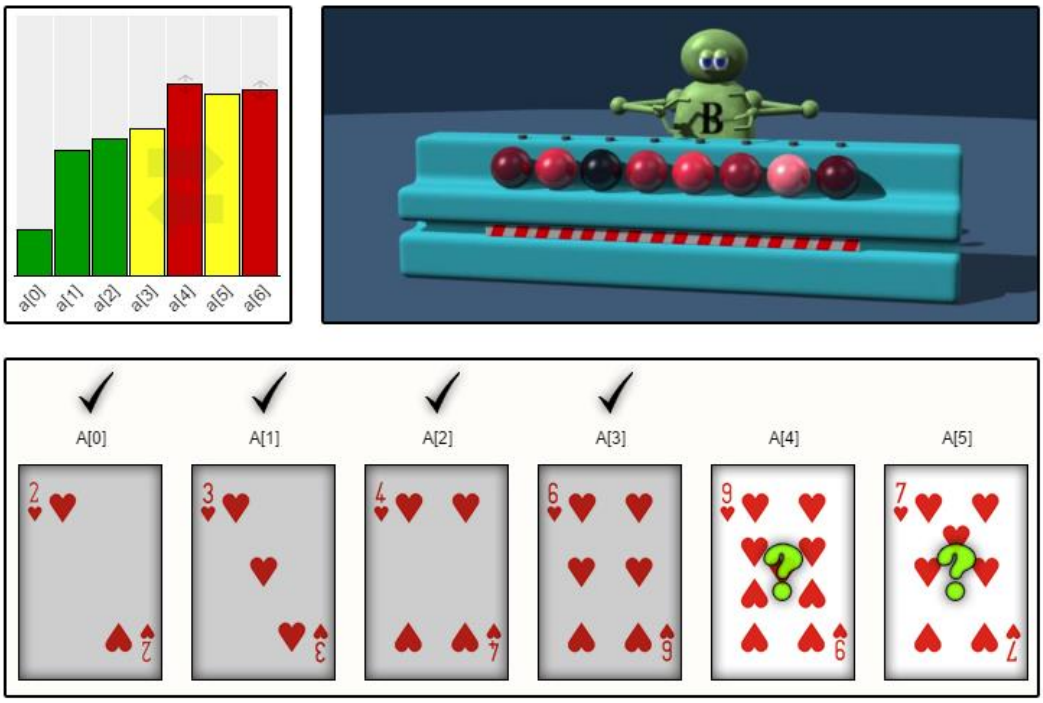
2. A második típusú animáció **mikro szinten, részletesen mutatja be** az algoritmus futása során elvégzett műveleteket kis adathalmazon (kb. 6-8 elem), az éppen végrehajtott művelet kiemelésével a pszeudokódban (Bernát, 2014; Hansen et al., 2002). Az általunk elkészített animációk közül ilyenek a rendezési algoritmusok oktatásánál felhasználható, 10. fejezetben tárgyalt interaktív animációk.
3. Harmadik típusú animációnak Hansen és társai olyan **makro-szintű animációt** javasolnak, amely nagy adathalmazon (kb. 50 elem) globálisan szemlélteti az algoritmust és bemutatja a hatékonyságával kapcsolatos jellemzőket. Az ilyen animációk a diákok mentális modelljeinek megerősítésére szolgálhatnak (Hansen et al., 2002).

A különböző típusú animációk más-más interaktivitást tartalmaznak, így a diákok különbözőképpen vehetnek részt a vizualizációs folyamatokban.

2.5.1 Az animációk külső megjelenítésével kapcsolatos javaslatok

- **A megfelelő modell választása.**

  Mivel az algoritmusok absztrakt adatszerkezetekkel és fogalmakkal dolgoznak, fontos a megfelelő modell megválasztása. Ez a modell általában az algoritmusban használt standard adatszerkezetet reprezentálja, pl. tömböt, gráfot, bináris fát (Esponda-Arguero, 2010; Fleischer & Kucera, 2002). Az alábbi ábrán tömböt reprezentáló modellek láthatók, melyeknél a tömbelemek értékei az oszlopok magasságaival, a golyók fényereivel, illetve kártyalapok értékeivel vannak kifejezve (Végh, 2011a; Végh & Stoffová, 2016, 2017).



11. ábra: Tömbelemek reprezentálása három különböző modell segítségével

Mivel az animációban használt modellek elemein általában valamilyen összehasonlításokat végzünk, ezért célszerű az elemek értékeit grafikusan kifejezni (pl. a fenti ábrán az oszlopok magasságával, golyók fényerejével, szívek számával a kártyalapokon) az egyszerű szöveges megjelenítés helyett (pl. egymás mellett elhelyezkedő, egyforma téglalapokba beírt számok) (Fleischer & Kucera, 2002).

- **Az algoritmus kisebb adathalmazon legyen szemléltetve.**

  Célszerű az algoritmusokat kisebb bemeneti adathalmazon bemutatni és elmagyarázni (Fleischer & Kucera, 2002). Az eddigi kutatások azt javasolták, hogy például egy tömb adatszerkezetet használó algoritmus működésének szemléltetéséhez 6–8 elemű tömböt érdemes használni. Nézetünk és tapasztalataink alapján ez elég az egyszerű algoritmusok működésének megértéséhez, de bizonyos tulajdonságaik felfedezésére kevés lehet. Némely algoritmusok egyes tulajdonságai csak több elem használatánál nyilvánulnak meg és válnak megfigyelhetővé.

- **A megfelelő adathalmaz választása.**

  Az algoritmusok jellegzetes vonásainak megjelenítéséhez fontos az animáció indításakor használt kezdeti adatok helyes megválasztása is. Sok esetben egy véletlen számokból álló adathalmaz megfelel az algoritmus első bemutatására. Ha azonban pl. a gyorsrendező algoritmust szeretnénk bemutatni, célszerű kezdetben

egy olyan 8 (vagy 16) elemből álló adathalmazt választani, amely a rendezési algoritmus futása során mindig két megközelítőleg egyforma méretű részre osztódik.

Javasolt a rendezési algoritmust alkalmazni már rendezett és ellenkező sorrendbe rendezett adatokra is. Az első esetben fontos, hogy meddig tart (hány összehasonlításra van szükség), míg az algoritmus felismeri, hogy az adatsorozat rendezett. A második esetben látni lehet, hogy az elemek kezdő pozícióinak távolsága a célpozíciótól a legnagyobb. Itt például ki lehet fejezni az algoritmus effektivitását a szükséges összehasonlítások számával.

- **Az animációhoz tartozzon magyarázat, amely segítségével a diákoknak könnyebb interpretálniuk az elemek grafikus reprezentációit és a végbemenő folyamatokat.**

  Az animációk segíthetik a különféle algoritmusok megértését, azonban fontos, hogy a diákok megértsék mi látható a vizualizáción, hogyan vannak az egyes elemek reprezentálva (pl. az oszlop magassága a programban használt változó értékétől függ), és hogy az animáció során éppen mi történik.

  Különféle kutatások azt sugallják, hogy ha az információk többféleképpen vannak kódolva (pl. animáció, forráskód és magyarázó szöveg formájában is), akkor azok könnyebben megérthetők. A magyarázat szerepelhet egyszerű szöveges formában az animáció előtt (pl. ha elektronikus tankönyvekbe integrált animációk esetén), vagy hang formájában (Mayer, 2009; T. L. Naps et al., 2002).

  Amennyiben szöveges formában jelenítjük meg az információkat a vizualizáció alatt, fontos hogy a diákoknak legyen elég idejük elolvasni és megérteni azokat. Mayer multimédia elvei alapján nem javasolt a szöveg és az animáció egyidejű megjelenítése, ugyanis a diákok nem képesek elolvasni, majd értelmezni a szöveget és közben figyelni az animációban végbement folyamatot is. Sokkal célszerűbb, ha a szöveg hanganyaggal (narrációval) helyettesíthető. Egy másik jó megoldás, ha a szöveg megjelenítése, majd elolvasása és megértése után egy gombnyomás segítségével indítható az animáció hozzá tartozó része (Mayer, 2009). A hang formájú magyarázat beiktatása elvégezhető fordított sorrendben is, amikor az animáció alatt indítható (szükség esetén) a hangszekvencia (Stoffová, 2004).

Nem feltétlenül szükséges, hogy a magyarázó szöveg az animáció része legyen. Egy egyetemi előadáson a magyarázat lehet a tanár szóbeli interpretálása, egy elektronikus tankönyvben a szöveges magyarázat szerepelhet közvetlenül a tananyagba beágyazott animáció előtt vagy után (Fleischer & Kucera, 2002).

Több kísérlet eredménye is felhívta arra a figyelmet, hogy az animációk nagyobb eredményességgel használhatók a tanulási folyamatban, ha azok nem önmagukban, hanem az elektronikus tananyag részeként szerepelnek (Kann et al., 1997; Kehoe et al., 2001; Rudder et al., 2007). Ilyenek lehetnek pl. hipertext szerkezetű, statikus diagramokkal és mintafeladatokkal gazdagított elektronikus tankönyvbe beágyazott animációk (Hansen et al., 2002). Az ilyen esetekben azonban rendkívül fontos, hogy a magyarázó szöveges részek és diagramok pontos összhangban legyenek az animációkban szemléltetett információkkal (pl. a gyorsrendezésben a vezérelem kiválasztása ugyanolyan módon történjen az animációban és a hozzá tartozó magyarázatban is). Ellenkező esetben az animációk nem segíteni fogják a diákok megértését, hanem össze fogják őket zavarni (T. Naps & Grissom, 2002).

- **Az algoritmus vizualizációja többféle nézetet tartalmazzon.**

  Az algoritmust az animáció során különbözőképpen lehet megfigyelni. Figyelhetünk a forráskódban az utasítások végrehajtásának sorrendjére, az elemek értékeinek változására az adatszerkezetekben, a ciklusváltozók értékeire, stb. Ezek miatt hasznos lehet, ha az algoritmus több különböző nézetben van megjelenítve a képernyőn (T. Naps & Grissom, 2002). Például, egyik részen láthatók az oszlopokkal szemléltetett tömbelemek, alatta a tömbelemeken végighaladó ciklusváltozók, mellette az algoritmus pszeudokódja az aktuális művelet kiemelt sorával/soraival.

- **A programkód vagy pszeudokód megjelenítése.**

  Az adott programozási nyelvhez tartozó forráskód vagy pszeudokód megjelenítése is segítheti a megértést. Fontos, hogy a kódban mindig ki legyen emelve az a programsor, amely éppen animálva van. Így a diákok össze tudják kapcsolni az animáció egyes eseményeit a programkódban található utasításokkal (Fleischer & Kucera, 2002; T. L. Naps et al., 2002).

  Egy konkrét programozási nyelvhez tartozó kódot csak programozás tanfolyamokon érdemes megjeleníteni. Az algoritmusok oktatásánál a konkrét

programozási nyelvekhez tartozó forráskódok megjelenítése helyett célszerűbb inkább a pszeudokódokat megjeleníteni, és azokban kiemelni az éppen végrehajtott folyamatokat. Így a diákok egyrészt absztrakt szinten, bármilyen programozási nyelvtől függetlenül foglalkoznak az algoritmusokkal, másrészt a diákok nem vesznek el a programkódok részleteiben, mivel a pszeudokódok magasabb szintű leírást tartalmaznak (Fleischer & Kucera, 2002).

- **Az animáció grafikailag legyen egyszerű, a vizualizációban használt színek és hangok is hordozzanak információkat.**

  Fontos olyan egyszerű grafikai elemek használata, amelyek nem vonják el a figyelmet a bemutatott algoritmusról. A felhasznált színeknek és hangeffektusoknak is legyen információtartalma (Fleischer & Kucera, 2002). Egy rendezési algoritmusnál a piros oszlopok jelenthetik a rendezetlen elemeket, a zöld oszlopok pedig a már helyükre került, rendezett elemeket. Így oldottuk meg pl. az oszlopok színeinek jelentését a saját alkalmazásainkban.

- **Az animáció tartalmazzon az algoritmus helyességével és hatékonyságával kapcsolatos információkat.**

  Hasznos, ha a diák valamilyen formában látja, hogy miért helyes és miért effektív az algoritmus (Fleischer & Kucera, 2002). Az algoritmusok bonyolultságának vizsgálata, hatékonyabb algoritmus keresése, hasznos lehet a különböző algoritmusok megértéséhez. Így pl. a rendezési algoritmusok vizsgálatánál és összehasonlításánál a haladó diákok számára hasznos lehet, ha láthatják az összehasonlítások és cserék számát is. Egy másik, még szemléletesebb megoldás, ha az egyes rendezési algoritmusok animációi párhuzamosan futnak egymás mellett (T. L. Naps et al., 2002). Erre találhatók példák a http://www.sorting-algorithms.com/ és a http://sorting.at/ weboldalakon.

- **Egy kurzuson belül használt animációk legyenek hasonlóak.**

  Egy algoritmusokat tárgyaló tanfolyam rengeteg vizualizációt foglal magában. A diákok számára hasznos, ha ezek ugyanolyan modelleket, nézeteket, irányító gombokat tartalmaznak. Amennyiben egy kurzuson belül az egyes algoritmus-animációk más-más külalakkal és kezeléssel rendelkeznének, a diákok számára több időre lenne szükség, míg az animációkban látottakat értelmeznék és az animációk kezelését elsajátítanák (Fleischer & Kucera, 2002).

2.5.2 Az animációk interaktivitásával kapcsolatos javaslatok

- **Az animáció irányítása legyen flexibilis.**

  A diákoknak legyen lehetőségük az animációk elindításán és leállításán túl előre vagy hátra lépni akár több lépést is az algoritmusokban (Bernát, 2014; Fleischer & Kucera, 2002; T. Naps & Grissom, 2002; T. L. Naps et al., 2002). Arra, hogy az animációt bármikor le lehessen állítani, többek között Mayer is felhívja a figyelmet, ugyanis nem mindegyik diáknak elegendő az egyes lépésekhez rendelt alapértelmezett időtartam ahhoz, hogy az adott lépést megfelelően átgondolja és megértse. Az animáció leállítására és folytatására szolgáló nyomógomb használatánál tökéletesebb megoldás, ha az egyes lépések után az animáció automatikusan leáll és a diák, miután átgondolta az adott lépést, egy gombnyomással tud továbblépni a következő lépés animálására. Így a diáknak nem kell arra is figyelnie, hogy mikor érdemes leállítani az animációt (Hansen et al., 2002; Mayer, 2009). Saját alkalmazásainkban sok esetben lehetőséget adtunk az algoritmus animálásának léptetéses lebonyolítására is.

- **Az animáció sebessége változó, vagy a felhasználó által módosítható legyen.**

  Egy adott algoritmus egyes részeinek animálása különböző sebességet igényel. Például, egy rendezési algoritmusban az összehasonlítások és a cserék a lényegesebb részek, ezeknél lassabb animálás szükséges (esetleg ezeknél a részeknél le is állhat az animáció), míg az algoritmus többi része lehet gyorsabb. Amennyiben a ciklusváltozókat követjük figyelemmel, a rendezési algoritmus első néhány végigfutása a lényeges a megértéshez, ezért a ciklusok további végigfutásainál már lehet gyorsabb az animálás (Fleischer & Kucera, 2002). Jó megoldás, ha az animáció sebességét a diák módosíthatja az igényeinek megfelelően. Saját alkalmazásainkban sok esetben lehetőség van az animálás sebességének beállítására és folyamatos irányítására a kísérlet alatt.

- **Az animációban legyen lehetőség saját bemeneti adatok megadására, az adatok módosítására.**

  A saját adatok megadása a diákokat arra bíztatja, hogy aktívabban vegyenek részt a vizualizáció folyamatában. Saját adatok megadásával a hallgatóknak lehetőségük nyílik kísérletezni, hiszen megfigyelhetik az algoritmus viselkedését különböző bemeneti adatokon (Fleischer & Kucera, 2002; Furcy et al., 2008; Hansen et al.,

2002; T. L. Naps et al., 2002). Több pedagógiai kísérlet eredménye is azt bizonyítja, hogy azon diákok, akik saját adatokat adhattak meg az animációkban, szignifikánsan jobb eredményeket értek el a teszteken (C. D. Hundhausen et al., 2002). Még érdekesebb lehet az adott algoritmussal való kísérletezés, ha a diákok nem csak a kezdeti adatokat adhatják meg, de azokat menet közben is módosíthatják ellenőrzött körülmények között.

- **Az animáció adaptálódjon a diák tudásszintjéhez.**

  A kezdőknek nehezebb lehet megérteni az animációt, ha túl részletes, sok információs ablaka van, vagy túl sok mindent lehet benne beállítani. Ők általában az előre definiált adatokkal szeretnek dolgozni. Ennek ellenére, a haladó diákoknak hasznos lehet, ha be tudják állítani saját adataikat, módosítani tudják a beállításokat, vagy részletesebb információt kapnak a folyamatokról (Fleischer & Kucera, 2002; T. L. Naps et al., 2002).

- **Az animáció közben hasznos lehet a hallgató kikérdezése.**

  Az animáció során feltett kérdések a diákokat arra ösztönzik, hogy jobban odafigyeljenek a vizualizációra (Fleischer & Kucera, 2002; Furcy et al., 2008; Hansen et al., 2002; T. L. Naps et al., 2002). Fontos azonban, hogy a kérdéseket mikor és milyen formában tesszük fel, hogy az ne terelje el a diákok figyelmét az animációban szemléltetett folyamatokról. A kérdéseket felteheti az animációs szoftver, lehetnek a diákoknak kiosztott papíron, elhangozhatnak szóban az előadás alatt, vagy lehetnek az animációt magába foglaló szövegkörnyezet részei (Grissom et al., 2003). Nem mindig szükséges, hogy a kérdéseket a diákoknak meg kelljen felelniük írásban vagy szóban. Sok esetben elég, ha a feltett kérdéssel elérjük, hogy a diák elgondolkodik rajta (Hansen et al., 2002). A kérdés feladata főleg az, hogy felkeltse a diák figyelmét és figyelmeztesse, mire kell emelt figyelemmel öszpontosítani az animációs kísérlet alatt.

- **Az animáció legyen szórakoztató.**

  A diákok könnyebben tanulnak, ha az animáció az elejétől a végéig érdekli őket. Nem célszerű ugyanazon, hosszú ideig tartó lépések sokszori megismétlése egymás után (Fleischer & Kucera, 2002).

  Rudder és társai játékelemeket iktattak be a valós életbeli szituációkkal szemléltetett animációba. Ilyen, a játékkal kapcsolatos aktivitások voltak például:

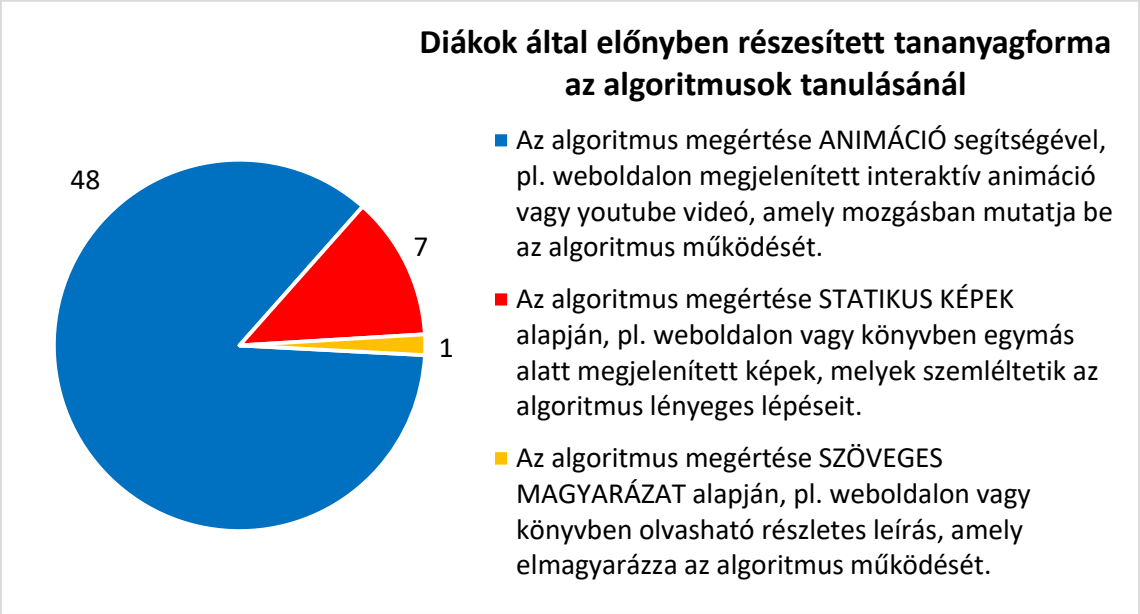
„találd meg a hibát”, „jósold meg a kimenetet”, „rendezd sorrendbe”. Mindezek fokozták a diákok kritikus gondolkodását (Rudder et al., 2007). Szórakoztatónak mondható az a videofelvétel is, ahol a rendezési algoritmusok tánc segítségével vannak bemutatva (Katai & Tóth, 2010).

3 A diákok véleménye az animációk használatáról az oktatásban

Mivel minden diáknak más a tanulási stílusa, egy egyszerű kérdőív segítségével kísérletet tettünk annak felmérésére, hogy egy új algoritmus megértésénél az általunk megkérdezett hallgatók (Selye János Egyetem informatika szakos hallgatói) közül mennyien részesítenék előnyben az animáció használatát a statikus ábrákkal vagy részletes szöveges magyarázattal szemben. A kérdőív a munka 1. mellékletében található.

A kérdőívet a Selye János Egyetem informatika szakos hallgatói töltötték ki a 2014/15-ös akadémiai év téli szemeszterében. A felmérés teljesen önkéntes volt, összesen 56 diák vett részt benne (26 elsős, 9 másodikos, 6 harmadikos, 6 negyedikes, 9 ötödikes).

A felmérésben egyetlen lényeges kérdést tettünk fel: „**Ha új algoritmust (pl. tömbelemek rendezése) kellene megtanulnod és az alábbi tananyagformák közül választhatnál, mit részesítenél előnyben?**”. Az diákok válaszát az alábbi grafikon összesíti.



12. ábra: Az egyes tananyagformákat választó diákok száma

Ez alapján elmondható, hogy a hallgatók szignifikáns többsége (85,7%) szívesebben tanulna algoritmusokat animációk segítségével, mint statikus képek vagy szöveges magyarázat alapján, $\chi^2(2, N = 56) = 70.107, p < 0.0005$ (SPSS Statistics, 2013) (Végh & Stoffová, 2016).

Ezzel sikerült bebizonyítanunk az 1. hipotézis érvényességét (a megkérdezett informatika szakos hallgatók többsége szívesebben tanulna informatikai algoritmusokat animációk segítségével, mint statikus ábrák vagy szöveges magyarázat segítségével).

4 Oktatás virtuális világokban

A doktori értekezéssel kapcsolatos kutatásunk következő fázisában a virtuális világokban történő oktatásra és ebben a világban kialakított interaktív oktatási modellekre összpontosítottunk.

A többfelhasználós virtuális környezetek (MUVE – Multi-User Virtual Environment) megjelenése óta több próbálkozás is történt ezek oktatásban való felhasználásukra. A modern, 3D, többfelhasználós virtuális környezetekben a felhasználók a saját maguk által megtervezett avatárjaik segítségével mozoghatnak, kommunikálhatnak, működhetnek együtt, fejezhetik ki érzéseiket, tanácsokat adhatnak egymásnak, virtuális világbeli objektumokat hozhatnak létre vagy módosíthatják azokat. Az egyik legelterjedtebb ilyen virtuális világ a Second Life (www.secondlife.com), melyet a világ különböző pontjain megközelítőleg 300-400 egyetem használ oktatási vagy kutatási célokra (Inman, Wright, & Hartman, 2010; Michels, 2010; Smith & Berge, 2009).

Az oktatásban a Second Life leggyakoribb és talán legelterjedtebb felhasználási módja a virtuális világbeli előadások és konferenciák szervezése a virtuális egyetemi kampusz tantermeiben. Ezek előnye, hogy a világ bármely pontjáról lehet csatlakozni, csupán a Second Life Viewer (kliens program) minimális hardverigényének megfelelő számítógépre és internetkapcsolatra van szükség. A virtuális világok azonban az ilyen fajta passzív információ befogadáson keresztüli tanuláson túl rengeteg modern, diák-központú tanulási lehetőséget is kínálnak aktív, kreatív, konstruktív és kollaboratív ismeretelsajátításra (Csízi & Végh, 2011; Kristóf, Végh, & Bodnár, 2011; Turcsányi-Szabó, Csízi, & Végh, 2012; Végh & Csízi, 2010).

Több kutatás eredménye is azt mutatja, hogy virtuális világok sikeresen használhatók a nyelvoktatásban, nyelvgyakorlásban, fényképészet, kereskedelem, programozás, vendéglátás és turizmus oktatásában, számítógépes animáció és grafika oktatásában (Micaela Esteves, Fonseca, Morgado, & Martins, 2009; M. Esteves, Fonseca, Morgado, & Martins, 2011; Joe Geigel, 2009; J. Geigel, 2010; Marešová, 2010; Penfold, 2008; Pereira, Martins, Morgado, & Fonseca, 2009; Roush, Nie, & Wheeler, 2009; Végh, 2014b; Wang & Burton, 2013).

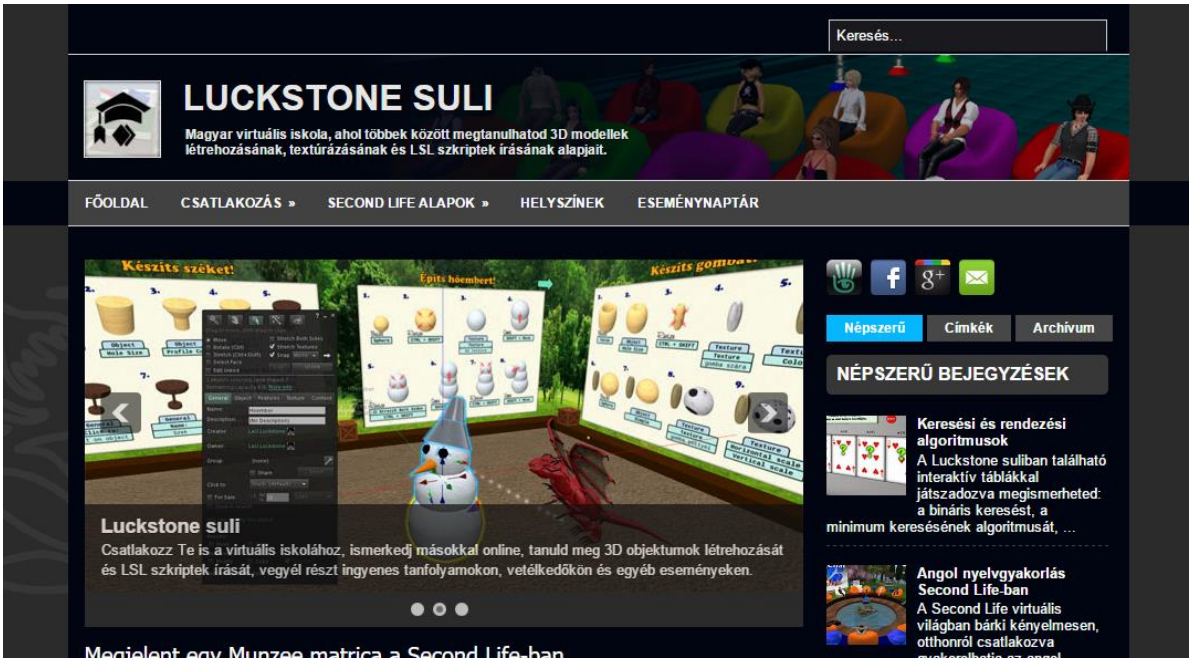
4.1 Virtuális iskola a Second Life-ban

A doktori munka keretén belül kialakítottunk egy oktatásra felhasználható teret a Second Life virtuális világban, ahol magyar nyelven kaphatnak információkat az érdeklődő diákok. A

virtuális iskola a „**Luckstone suli**” nevet kapta, amely az avatárunk nevéből eredt. A kialakított tér a virtuális világban a **secondlife://Hippoden/92/132/27** címen érhető el.

13. ábra: A „Luckstone suli” virtuális iskola főépülete és a mellette található homokozó (sandbox)

Az iskolához létrehoztunk egy weboldalt is, amely különféle leírásokat, helyszínek listáját és egy eseménynaptárt is tartalmaz: http://luckstonesuli.blogspot.sk/.



14. ábra: A „Luckstone suli” virtuális iskola weboldala

A virtuális iskola kialakításához különböző szempontokat vettünk figyelembe, ezek részletes leírása megtalálható a (Turcsányi-Szabó et al., 2012; Végh, 2012) publikációkban. A célunk egy olyan tér kialakítása volt, ahol önállóan is elsajátíthatók többek között a virtuális

világbeli modellek létrehozása, majd ezekhez az interaktivitást bebiztosító szkriptek írása, azonban tanár vezetésével is végezhetők csoportos aktivitások.

A virtuális térben több különálló helyszínt alakítottunk ki (Végh & Turcsányi-Szabó, 2017):

- **Alap információkat tartalmazó helyiség (földszint)**

  Az ide látogató felhasználók erre a helyre érkeznek. Itt különböző információs táblák találhatók: Second Life csoporthoz csatlakozás, Facebook csoporthoz csatlakozás, iskola weboldalának megtekintése, üzenet küldése számunkra, iskolával kapcsolatos kérdőív kitöltése, eseménynaptár.

  Továbbá itt találhatók hivatkozások (teleport táblák), amelyekre kattintva a felhasználó gyorsan eljuthat az iskola különböző helyszíneire.

- **Segítség a virtuális térben való mozgás elsajátításához (első emelet)**

  A kezdő felhasználóknak 11 információs tábla mutatja be az avatárjuk és a kamera mozgatásához szükséges ismereteket, ill. a felhasználói felület fontosabb nyomógombjainak leírását. Itt a felhasználók kaphatnak egy jegyzetlapot is, amely segítségével 3D modellek létrehozásának első lépéseit ismerhetik meg.

- **Beszélgető kör és logikai játékok (második emelet, tetőtér)**

  Kikapcsolódásra, beszélgetésre kialakított tetőtéri helyiség kör alakban elrendezett ülőhelyekkel. A kör közepén memóriajáték, szám- és képkirakó játékok találhatók.

- **Homokozó (sandbox)**

  Az iskola épülete mellet kialakított üres terület, ahol a diákok kedvükre kipróbálhatják az építkezést.

- **Tanterem (skybox, a virtuális térben 240 m magasságban található)**

  Csoportos órák, előadások, kvízjáték részére kialakított helyiség, amelyben a tanterem különféle elrendezése választható ki: aktivity játék (18 ülőhely), kvíz játék (16 ülőhely), modellezés és szkriptek létrehozásának oktatása (10 diák + 1 tanár ill. 6 diák + 1 tanár számára kialakított ülőhelyek, ahol minden résztvevő külön építési területtel rendelkezik), magánóra (1 diák + 1 tanár számára kialakított ülőhelyek), előadás (30 diák + 1 tanár számára kialakított ülőhelyek), társalgó (20 diák + 1 tanár számára, kör formában kialakított ülőhelyek), üres helyiség.

A tanteremben különböző táblatípusok választhatók ki: prezentáció (a tanár saját előadásának képeit jelenítheti meg), böngészés (beépített internet böngésző segítségével a felhasználók közösen barangolhatnak különböző webhelyeken), rajzolás (a tanár és a résztvevők tetszés szerint rajzolhatnak a táblára).

- **3D objektumok készítése (skybox, a virtuális térben 3000 m magasságban található)**

  Egy újabb homokozó (sandbox), amely körül információs táblák és gyakorló feladatok segítik az építkezés alapjainak elsajátítását. A helyiség 45, fokozatosan nehezedő témakör leírását tartalmazza 5 információs táblán (mindegyiken 8 témakörrel). Ezen kívül a diákok rendelkezésére áll 9 faragott alapobjektum (sculpted prim) létrehozásához szükséges textúra, és a diákok modelljeinek textúrázására szabadon felhasználható további 144 kép 16 különböző kategóriában. A helyszínen, 6 objektum elkészítésének pontos lépéseit bemutató információs táblák is találhatók (szék, hóember, gomba, varázskalap, fagylalt, gyertya), melyekhez az általunk előre elkészített textúrák, animációk és szkriptek szintén elérhetők a diákok számára.

- **Programozás, szkriptek létrehozása (skybox, a virtuális térben 3200 m magasságban található)**

  Egy másik homokozó (sandbox), amely körül információs táblák segítik szkriptek írásának elsajátítását. A helyiség 56, fokozatosan nehezedő témakör leírását tartalmazza 7 információs táblán (mindegyiken 8 témakörrel). Az érdeklődők számára 43, előre elkészített szkript is rendelkezésre áll.

- **Keresési és rendezési algoritmusok interaktív modelljei (skybox, a virtuális térben 2800 m magasságban található)**

  A helyiség 6 informatikai algoritmust bemutató interaktív modellt tartalmaz: bináris keresés, minimum keresése, beszúró rendezés, kiválasztó rendezés, buborékrendezés, egyszerű cserés rendezés. Az itt található modellekkel részletesebben a 4.3 és 4.4 fejezetekben foglalkozunk.

- **Angol nyelv gyakorlására kialakított helyiség (skybox, a virtuális térben 3400 m magasságban található)**

  Egy kisebb helyiség, amely 9, kör formában elrendezett ülőhelyet tartalmaz, közepén egy angol nyelvű szókitaláló játékkal. A terem tartalmaz továbbá 2 memóriajátékot állatok, zöldségek és gyümölcsök angol neveinek begyakorlására, különböző virtuális világbeli helyek címeit csoportos kirándulás céljára, és egy naptárt ingyenes nyelvgyakorló Second Life-beli helyek eseményeivel.

A kialakított virtuális térben többfajta oktatási és kutatási tevékenység is megvalósult:

- 3D grafika és programozás oktatása (Gubo, Jaruska, & Végh, 2012; Végh, 2014b),
- Keresési és rendezési algoritmusok interaktív animációs modelljeinek használata (Végh, 2014a; Végh, Gubo, & Takáč, 2015).

Az általunk létrehozott virtuális területet a Selye János Egyetem tanári informatika szakos hallgatói is kihasználták szakdolgozataik elkészítéséhez, melyek keretén belül a virtuális iskolában oktatási tevékenységet is végeztek az alábbi témákban:

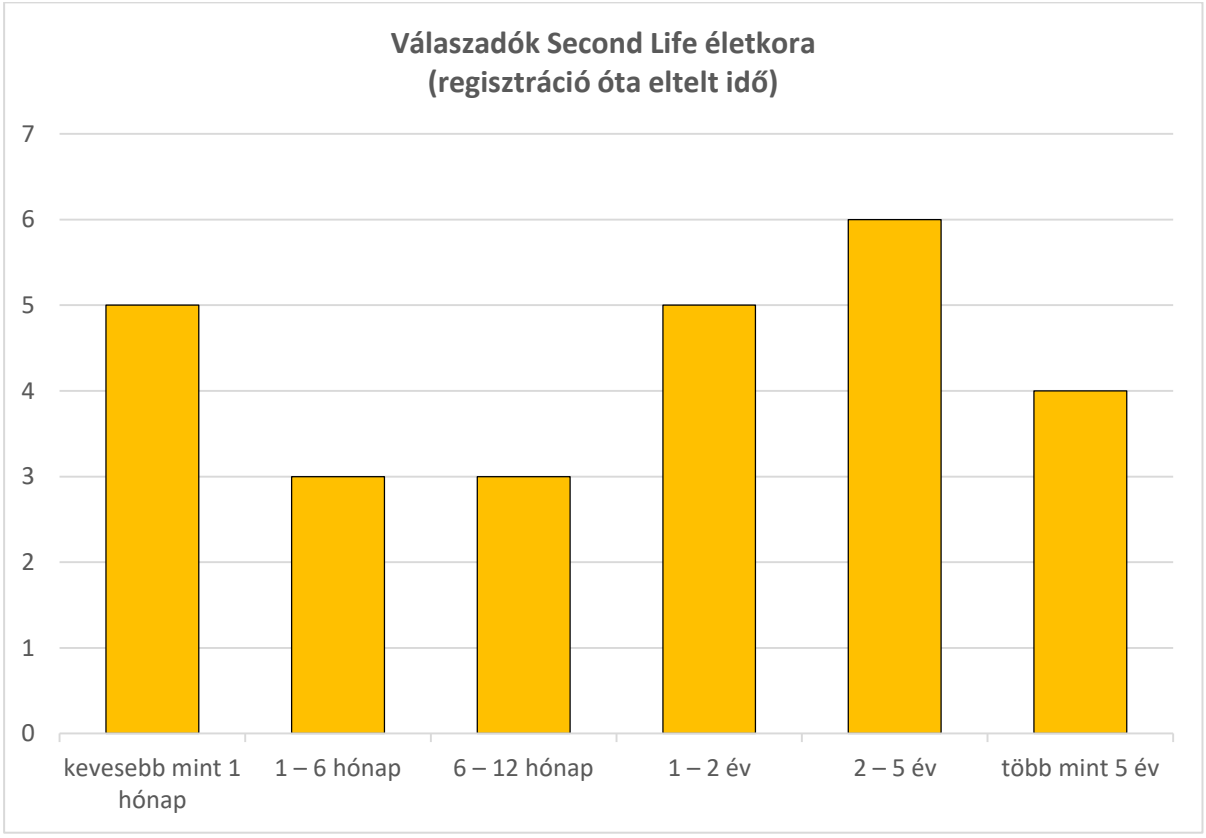
- 3D modellek készítésének oktatási lehetőségei a virtuális világokban (Sülei, 2013),
- Angol nyelv oktatásának lehetőségei virtuális világokban (Kiss, 2014).

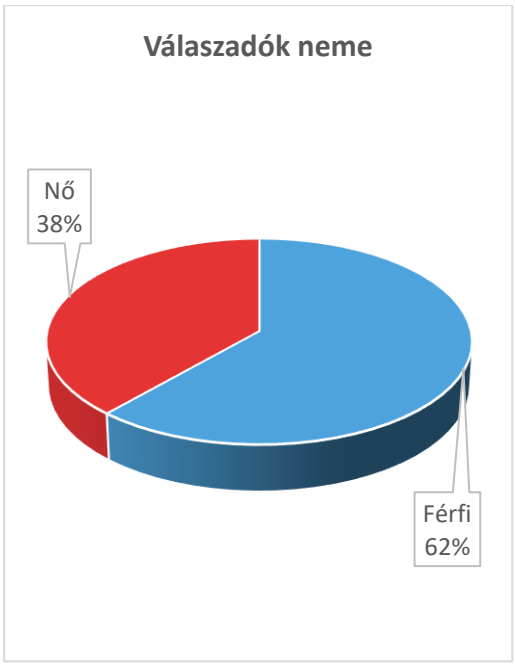
A virtuális iskola területén elhelyeztünk egy információs táblát is, amelyre kattintva a felhasználók kitölthettek egy kérdőívet (Google űrlap). A kérdőív kérdései a doktori értekezés 2. mellékletében találhatók. Mindenekelőtt arra voltunk kíváncsiak, kik látogatják a virtuális iskolát, milyen tevékenységeket szoktak végezni a virtuális világban, milyen 3D modellezéssel és programozással kapcsolatos tapasztalataik vannak, és mit tanulnának szívesen ebben a virtuális térben.

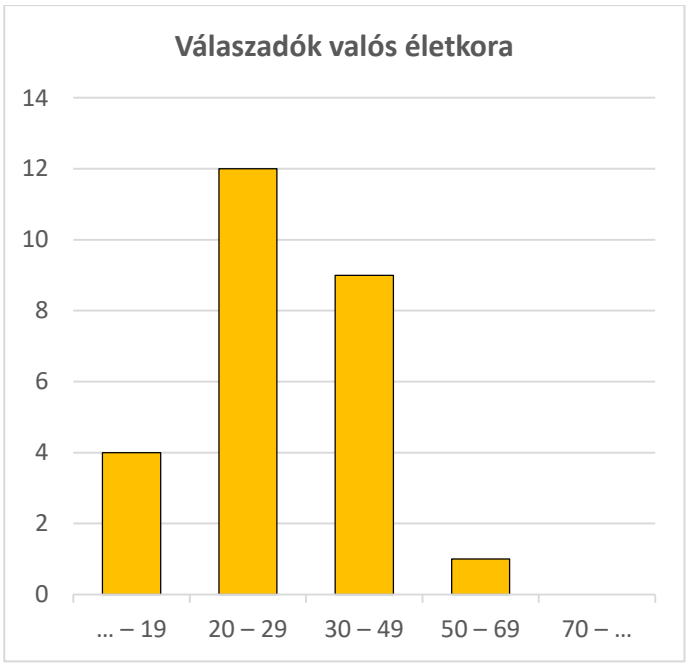
4.2 Virtuális iskolával kapcsolatos kérdőív kiértékelése

A kérdőívet 2014. január és 2016. december között összesen 26 Second Life felhasználó töltötte ki. A felhasználók nem voltak semmilyen feltételek alapján kiválogatva, a virtuális iskola területére ellátogató felhasználók voltak. A kérdőív kitöltését önként vállalták a virtuális iskola területére kihelyezett táblára kattintva (Végh, 2016d; Végh & Turcsányi-Szabó, 2017).

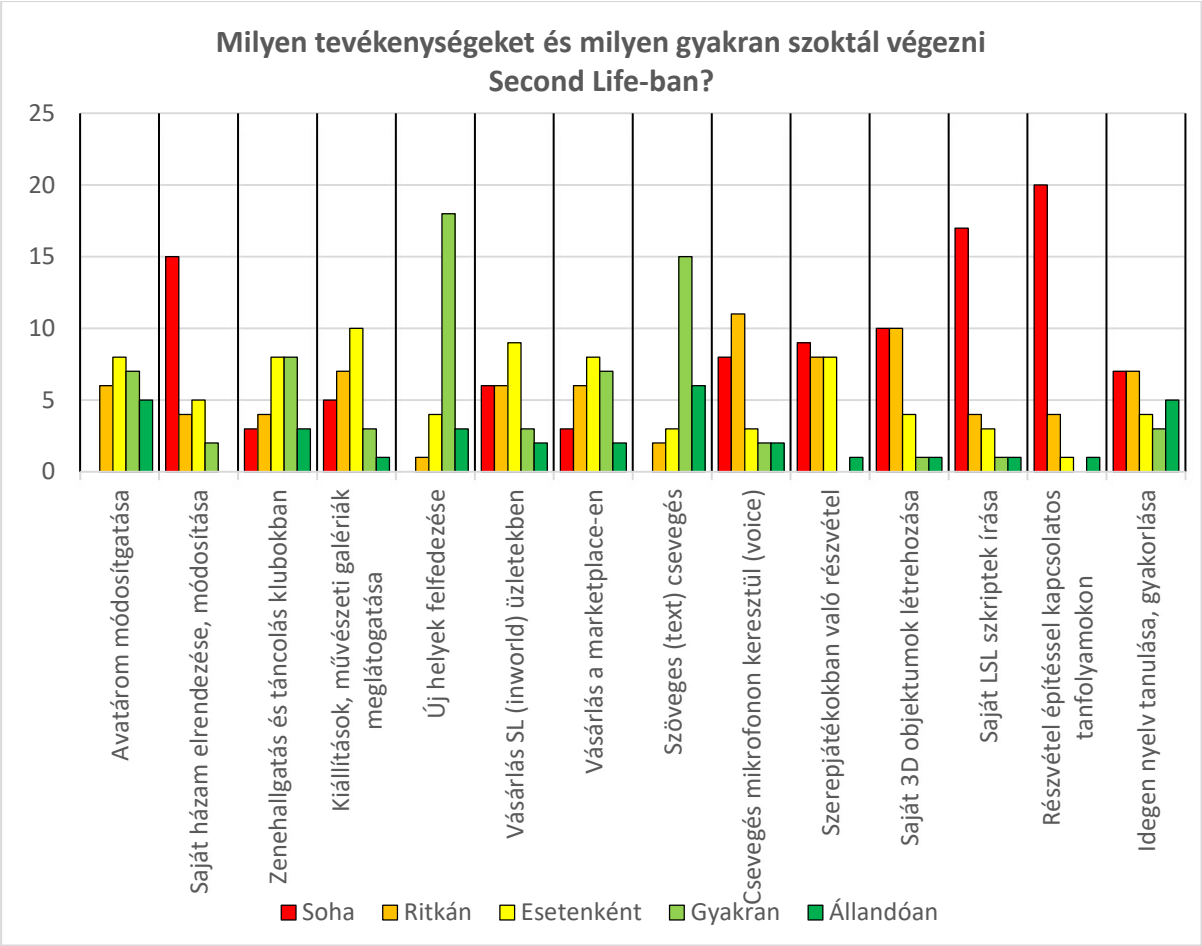
Az alábbi grafikonok szemléltetik a kérdőívre adott válaszokat.



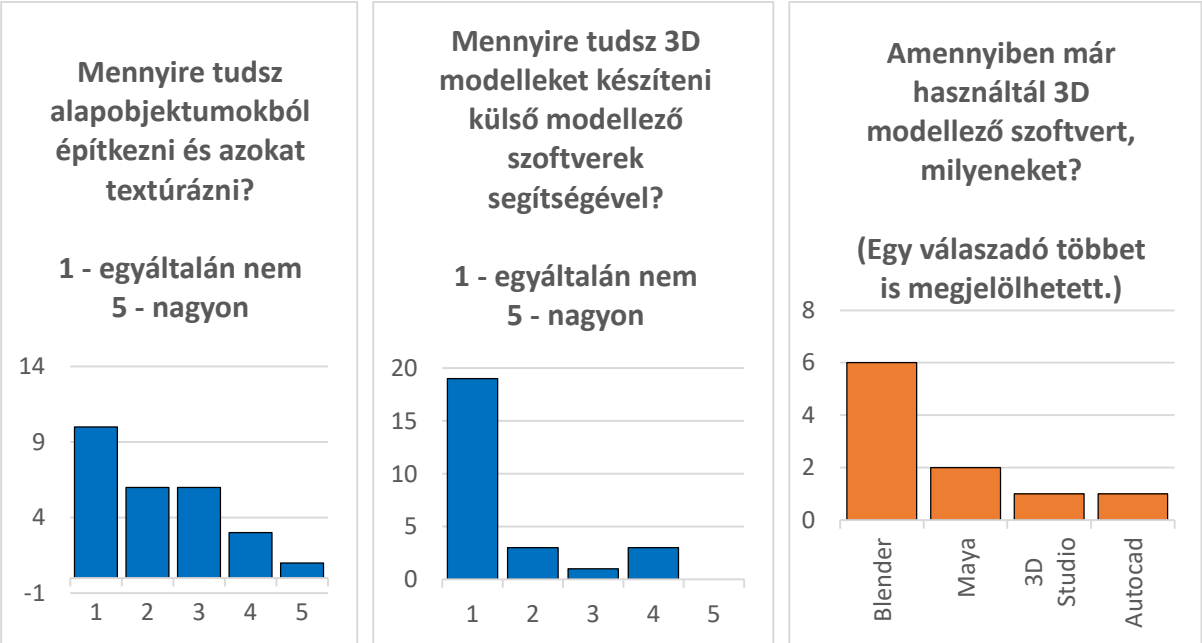




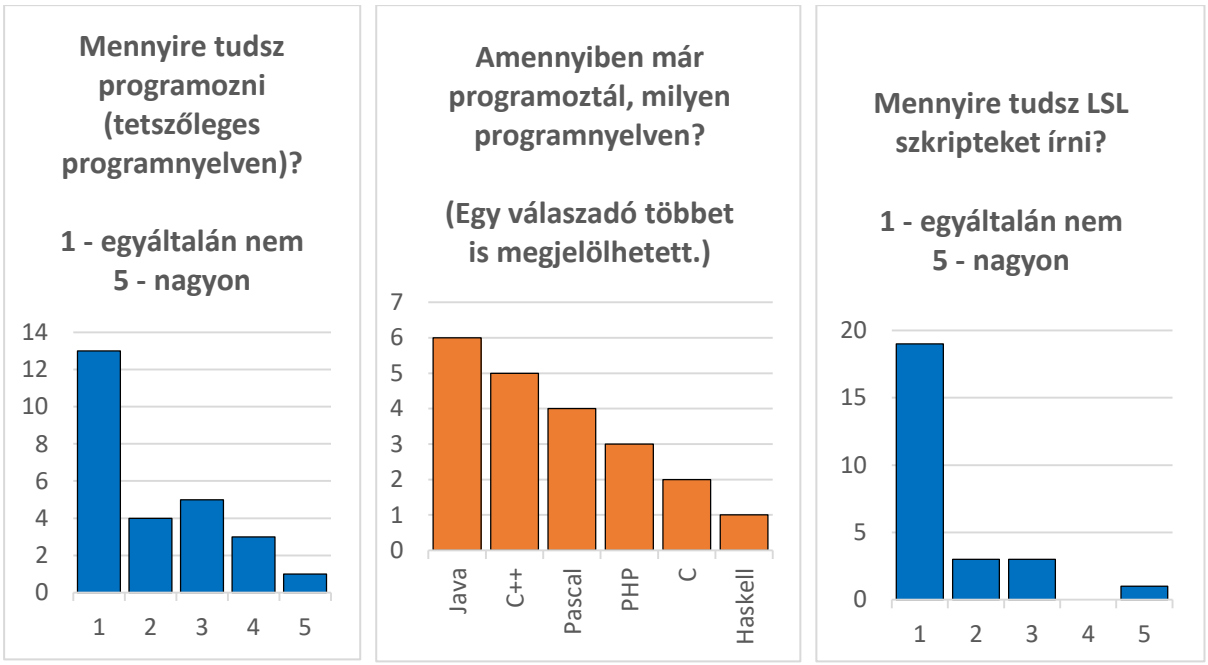
15. ábra: A virtuális iskolával kapcsolatos kérdőív kérdéseinek kiértékelése – demográfiai adatok



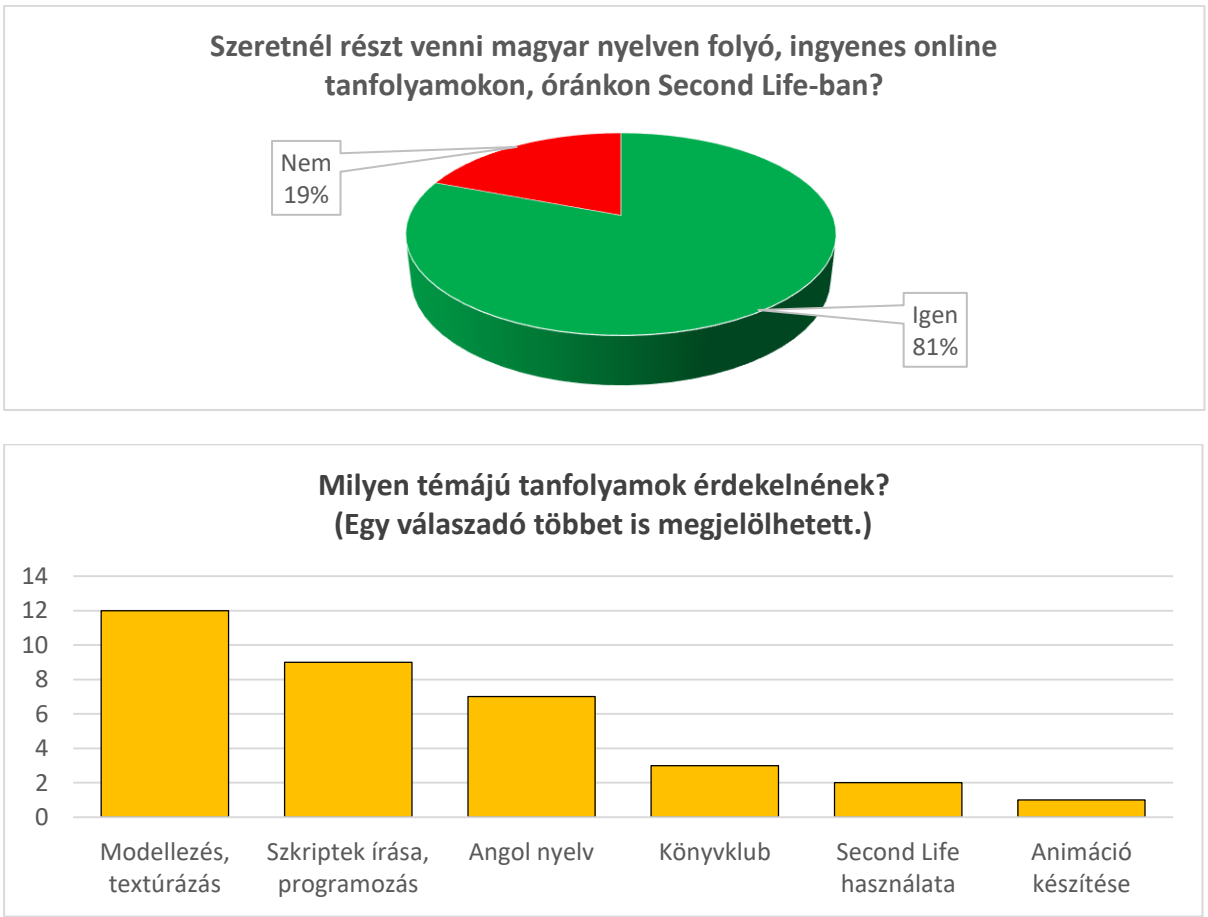
16. ábra: A virtuális iskolával kapcsolatos kérdőív kérdéseinek kiértékelése – Second Life-ban végzett tevékenységek kimutatása



17. ábra: A virtuális iskolával kapcsolatos kérdőív kérdéseinek kiértékelése – 3D modellezésben való jártasságok kimutatása



18. ábra: A virtuális iskolával kapcsolatos kérdőív kérdéseinek kiértékelése – programozásban való jártasságok kimutatása



19. ábra: A virtuális iskolával kapcsolatos kérdőív kérdéseinek kiértékelése – diákok jövőbeli érdeklődési körének kimutatása

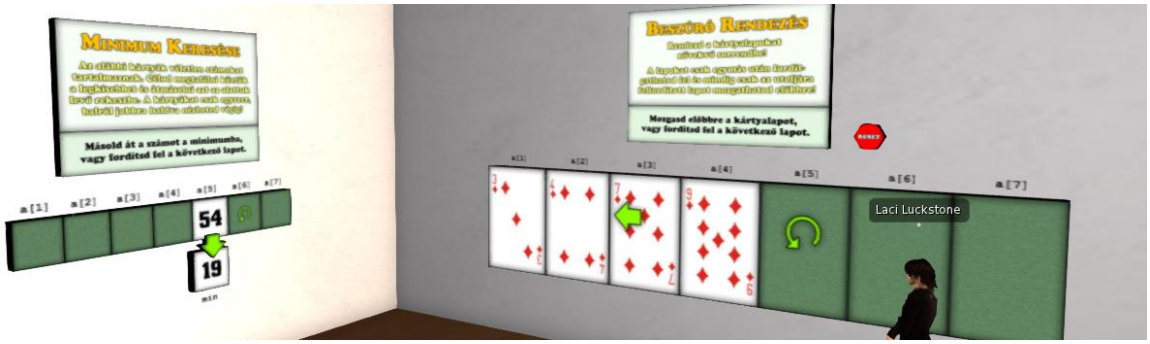
A válaszok közül főként az érdekelt bennünket, hogy mennyien szeretnének modellezést vagy programozást tanulni Second Life környezetben, mivel e két ismeret segítségével lehet a virtuális világban bármilyen új objektumot létrehozni.

A 26 megkérdezett felhasználó közül csupán 15-en válaszoltak arra a kérdésre, hogy milyen tanfolyamok érdekelnék őket. Közülük 14-en (93%) sorolták fel a válaszukban a modellezést vagy a programozást, és csak 1 (7%) felhasználó válaszában nem szerepelt se a modellezés, se a programozás tanfolyam.

4.3 Keresési és rendezési algoritmusok virtuális világbeli modelljeinek bemutatása

Mivel a doktori munkánk a programozás oktatásán belül elsősorban az algoritmusok interaktív animációinak oktatási célú felhasználásával foglalkozik, a kutatás kezdeti fázisában létrehoztuk néhány keresési és rendezési algoritmus virtuális világbeli modelljét, majd megpróbáltuk felmérni, hogy a felhasználóknak mennyire sikerült elsajátítaniuk ezek használatát.

A létrehozott animációs modellek gyűjteménye a **secondlife://Hippoden/124/135/2800** címen érhetők el a Second Life virtuális világban. A gyűjtemény az alábbi keresési és rendezési algoritmusokból áll: bináris keresés, minimum keresése, beszúró rendezés, kiválasztó rendezés, buborékrendezés, egyszerű cserés rendezés.



20. ábra: Second Life virtuális világban létrehozott interaktív animációs modellek: minimum keresése és beszúró rendezés

Az interaktív animációkat ahol lehetett, úgy próbáltuk meg kialakítani, hogy az egyes feladatokat a felhasználók csupán a véletlenre hagyatkozva ne tudják megoldani. Például, a bináris keresésnél nem generáltunk ki előre számokat, hanem azokat a lapok felfordításakor folyamatosan határoztuk meg. Így, bár a felhasználónak úgy tűnhet, hogy teljesen véletlen

számokkal dolgozik, mégis csak a megfelelő algoritmus alkalmazásával tudja megoldani a feladatot.

Az ilyen interaktív modellek létrehozásához a virtuális térben mindenekelőtt a 3D modell megtervezése szüksége. Ez a Second Life környezetben háromféleképpen valósítható meg (Gubo et al., 2012; Végh, 2012):

- a virtuális világban rendelkezésre álló alapobjektumok (primek) segítségével,
- faragott (sculptured) alapobjektumok létrehozásával, külső szoftver segítségével,
- 3D modellező szoftverben elkészített modell importálásával.

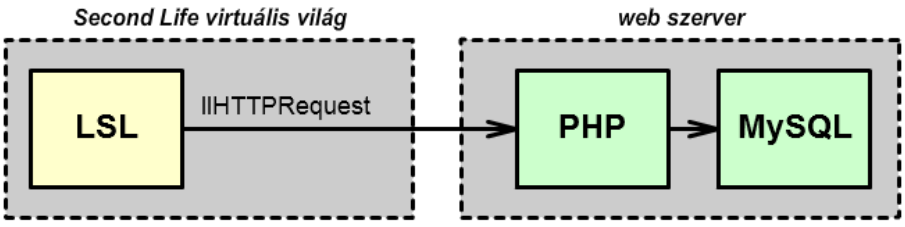
A modellek elkészítéséhez a harmadik módszert választottuk, a keresési és rendezési algoritmusok modelljeit Blender-ben készítettük el, majd a textúrákkal együtt importáltuk a virtuális környezetbe.

Ahhoz, hogy a modellek reagáljanak az eseményekre a virtuális térben, LSL (Linden Scripting Language) szkriptek írása szükséges. Az LSL szkript segítségével különféle módon animálhatók az objektumok és azok részei (Végh, 2014a):

- objektumok mozgatásával, méretezésével, elforgatásával,
- objektumok textúráinak módosításával, animálásával,
- az objektum egyes részeinek eltüntetésével és megjelenítésével.

A keresési és rendezési algoritmusok interaktív animációinak elkészítéséhez ezek közül az első két módszert használtuk.

Ahhoz, hogy a felhasználók lépéseit el tudjuk menteni későbbi elemzésre, létrehoztunk egy MySQL adatbázist, melyet összekapcsoltuk a virtuális világbeli objektumainkkal. Az adatokat az LSL szkriptnyelv llHTTPRequest utasításának segítségével küldtük el egy külső web szerveren futó PHP szkriptnek, amely azokat elmentette egy MySQL adatbázisba.



21. ábra: Adatok mentésének módja MySQL adatbázisba

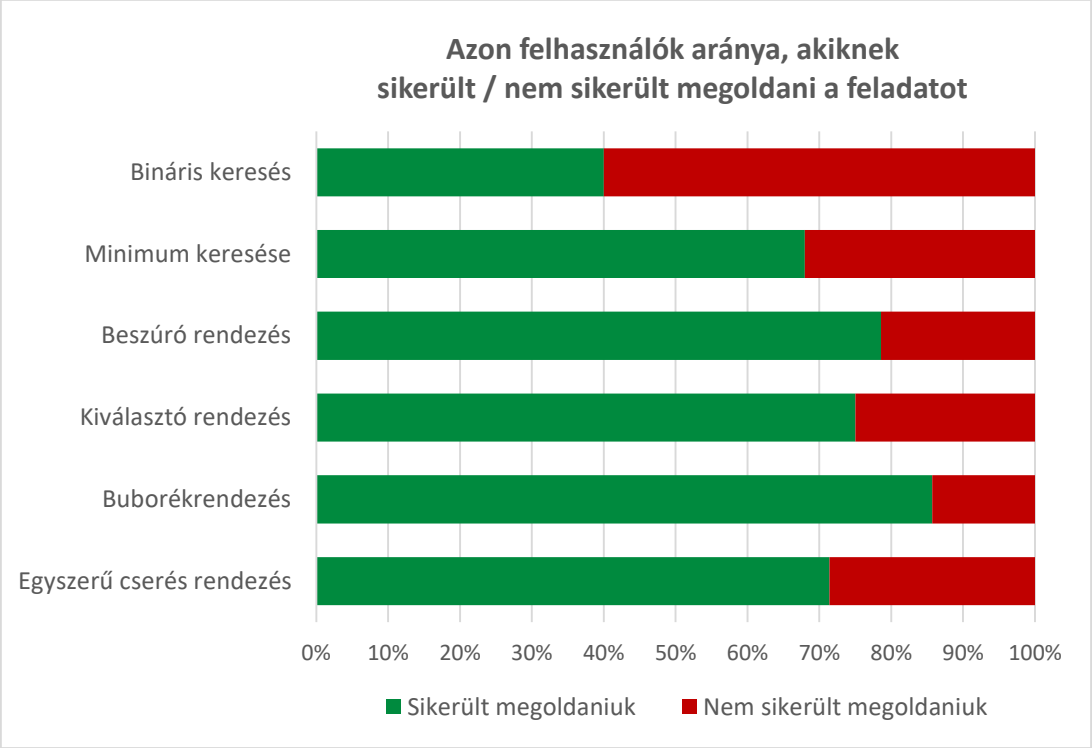
4.4 Keresési és rendezési algoritmusok modelljeivel végzett kísérletek eredményeinek kiértékelése

Az előzetes felméréshez kialakított interaktív animációkat 52 különböző Second Life felhasználó próbálta ki 2013. augusztus és 2015. december között. A felhasználókat nem válogattuk semmilyen szempontok alapján, bárki használhatta az interaktív objektumokat, aki ellátogatott a Second Life-ban kialakított virtuális területre. Az egyes animációkat kipróbáló felhasználók száma az alábbi táblázatban látható.

	Bináris keresés	Minimum keresése	Beszúró rendezés	Kiválasztó rendezés	Buborék-rendezés	Egyszerű cserés rendezés
Felhasználók száma	**30**	**25**	**14**	**20**	**7**	**7**

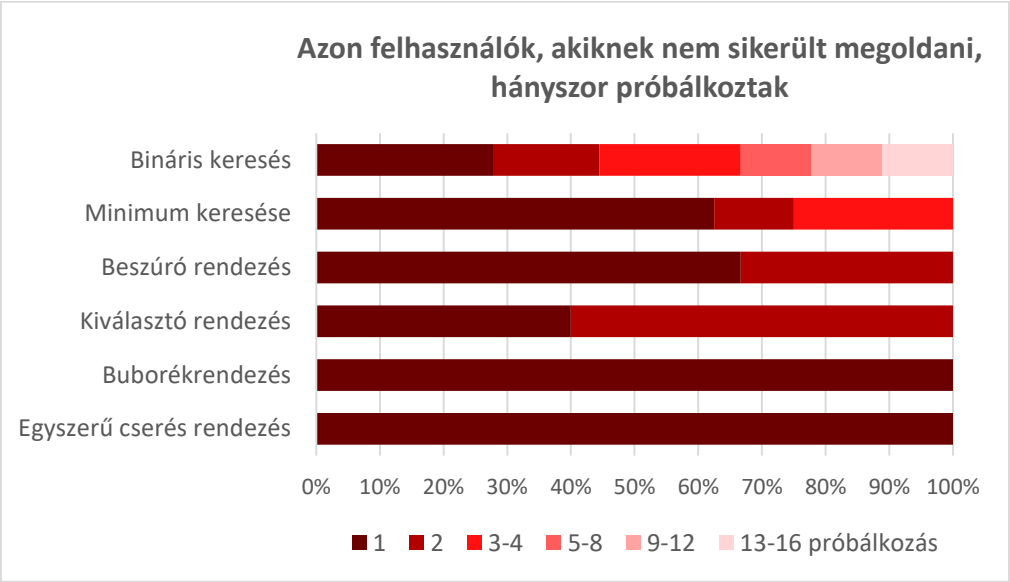
2. táblázat: Az animációkat kipróbáló Second Life felhasználók száma

Az animációk segítségével megoldandó feladat egy-egy szám megkeresése (bináris keresés, minimum keresése), vagy a kártyalapok növekvő sorrendbe való rendezése volt a megfelelő algoritmusokat használva (beszúró rendezés, kiválasztó rendezés, buborékrendezés, egyszerű cserés rendezés). A megoldások sikerességét az alábbi grafikon szemlélteti.



22. ábra: Sikeres és sikertelen megoldók aránya

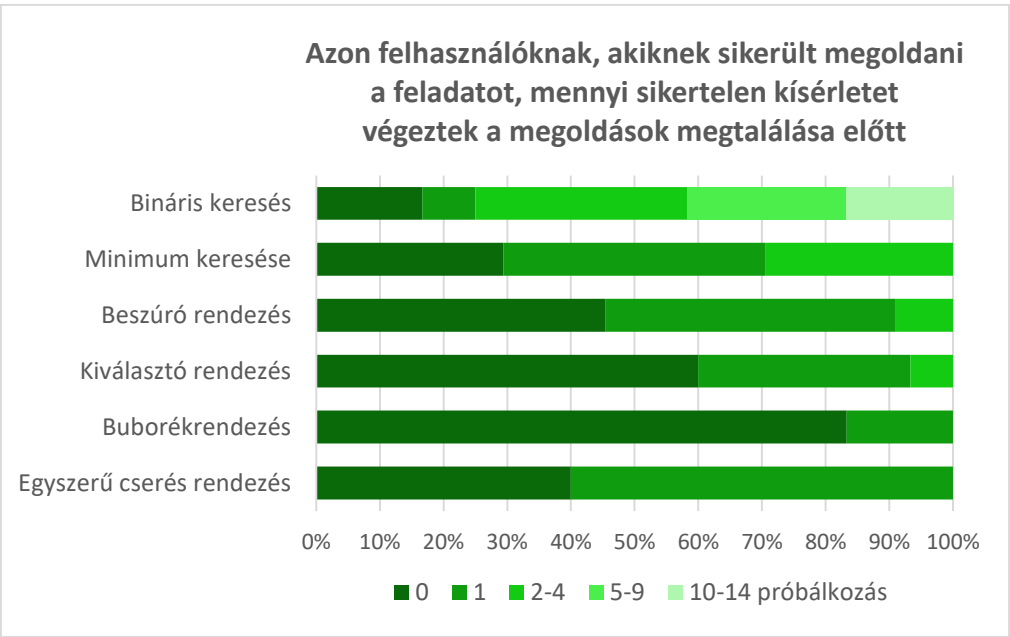
Az alábbi grafikonon látható, hogy a sikertelen megoldók mennyiszer próbálkoztak a feladatok megoldásával.



23. ábra: Próbálkozások száma a sikertelen megoldók esetében

A grafikonból kiolvasható, hogy a sikertelen megoldók többsége csak egyszer vagy kétszer próbálkozott, valószínűleg csak kipróbálták az interaktív objektumok használatát, de nem keltette fel az érdeklődésüket.

Az alábbi grafikon azt szemlélteti, hogy a sikeres megoldók mennyi sikertelen kísérletet végeztek a helyes megoldások megtalálása előtt.



24. ábra: A sikertelen próbálkozások száma a megoldások megtalálása előtt

A grafikonból látható, hogy a sikeres megoldók többségének néhány (0-4) sikertelen kísérlet elég volt az animációkkal való megismerkedésre és a feladatok megoldásainak megtalálására.

2015 elején úgy döntöttünk, hogy az itt bemutatott rendezési algoritmusok animációit elkészítetjük weboldalba beágyazható formában is (HTML5 és JavaScript használatával), így azok jóval szélesebb körben elérhetővé válnak (a felhasználóknak nem kell Second Life-ba regisztrálniuk, majd elsajátítaniuk a virtuális világ kezelését) és gyengébb számítógépen is probléma nélkül használhatók (a Second Life futtatása erősebb grafikus kártyát igényel).

Az új animációk keresztülmentek kisebb módosításokon is, elsősorban az irányításukat tekintve. Az animációk szigorúbban követik a rendezési algoritmusok egyes lépéseit, így jobban felhívják a felhasználók figyelmét azokra. A módosított animációkat, az elsős informatika szakos hallgatók körében végzett pedagógiai kísérlet bemutatását, ill. az eredmények elemzését a munka 7. fejezete tartalmazza.

4.5 Tapasztalatok, módszertani tanácsok

A virtuális világok használatának az oktatásban több előnye, de sajnos jelenleg még hátránya is van. Ezek közül néhányat az alábbi táblázatban foglaltunk össze.

Előnyök	Hátrányok
• **Motiváló 3D környezet** – mivel hasonlít a modern játékok környezetére, a diákok számára rendkívül vonzó. Szívesen készítenének akár saját objektumokat is ebben a környezetben, tehát a modellezés és programozás oktatására ilyen szempontból nagy előnnyel rendelkezik. • **Hozzáférés otthonról is** – a modellek építését és programozását bárhonnan el lehet végezni, csupán megfelelő számítógépre és internetkapcsolatra van hozzá szükség. • **Kollaboráció** – a virtuális világbeli modellek tervezésében és szkriptelésében a diákok együttműködhetnek. A virtuális térben lehetőségük van mikrofonon keresztül is kommunikálni. • **Megosztás** – az elkészített objektumokat, szkripteket, animációkat, szöveges	• **Magas technikai követelmények –** a 3D környezet használatához erősebb grafikus kártya és nagyobb sávszélességű internetkapcsolat szükséges. • **Saját kliens program, regisztráció** – a Second Life virtuális környezet nem futtatható internet böngészőben, használatához kliens program letöltése és telepítése szükséges. A virtuális térbe a belépés regisztrációhoz kötött, melynek egyik feltétele a 16. életév betöltése. • **A környezet kezelésének elsajátítása hosszabb időt is igénybe vehet** – a virtuális térben az alapvető mozgás, kamera és mikrofon kezelése, objektumokkal való interaktivitás, saját ruhatár és tárgyak használatának elsajátítása több órát is igénybe vehet. • **Az oktatásra használható interaktív objektumok létrehozása technikailag korlátozott** – az oktatásra használható

dokumentumokat, képeket a diákok könnyen megoszthatják egymással. • **Anonimitás** – mivel a diákok az avatárok segítségével vannak reprezentálva a virtuális térben és sokszor személyesen nem is ismerik egymást, bátrabban kapcsolódnak be a tevékenységekbe (pl. idegen nyelv gyakorlásakor a félénkebbek is aktívabban bekapcsolódnak a beszélgetésekbe).	interaktív animációs 3D modellek létrehozása egyrészt bonyolultabb mint a weboldalakon megjeleníthető 2D modelleké, másrészt jelenleg még eléggé korlátozott is (pl. a földterület méretétől függ hogy mennyire bonyolult modell helyezhető ki a virtuális térbe, nem lehetséges az összefüggő 3D modell formáját módosítani, sík felület textúrájára sem lehet dinamikusan rajzolni vagy szöveget kiírni). • **LSL szkriptnyelv** – a nyelv még jelenleg is folyamatos fejlődésen megy keresztül, új függvények jelennek meg és néhány függvény elavulttá válik. Továbbá, nem találhatók meg benne más nyelvekből megszokott adatszerkezetek, pl. tömb, halmaz, struktúra (helyettük listák használhatók). Így a programozás oktatására csak kellő körültekintéssel használható.

3. táblázat: Virtuális világokban való oktatás előnyei és hátrányai

Az algoritmusok és a programozás alapjainak oktatása a virtuális világokban kétféleképpen valósulhat meg:

1. **Saját LSL szkiptek írásával** – A diákok saját szkripteket írnak az LSL (Linden Scripting Language) eseményvezérelt programozási nyelvet használva.

   Az ilyen fajta programozás oktatás nagy motiváló hatással lehet, hiszen a szkriptek segítségével virtuális világbeli háromdimenziós objektumokat tehetnek interaktívvá, majd később az objektumokat megoszthatják másokkal, vagy akár kitehetik a Second Life piacterére (https://marketplace.secondlife.com/).

   Az LSL nyelv használata a programozás oktatására azonban több hátránnyal is jár. A nyelv jelenleg még folyamatosan fejlődő, időről időre új függvények jelennek meg benne vagy vállnak elavulttá. További hátránya, hogy túl sok, különböző függvény létezik a 3D objektumok manipulálására és textúrázására, bár megfelelő tananyag összeállításával ezek közül a fontosak fokozatosan elsajátíthatók. A nyelv tartalmaz minden lényeges vezérlési szerkezetet (feltételvizsgálatot, számláló-, előtesztelő- és utótesztelő ciklusokat), azonban több olyan fontos adatszerkezet nem található meg benne, amelyek más, kezdő programozók számára oktatott programozási nyelvben előfordul. Ilyen például a tömb, rekord/struktúra, halmaz

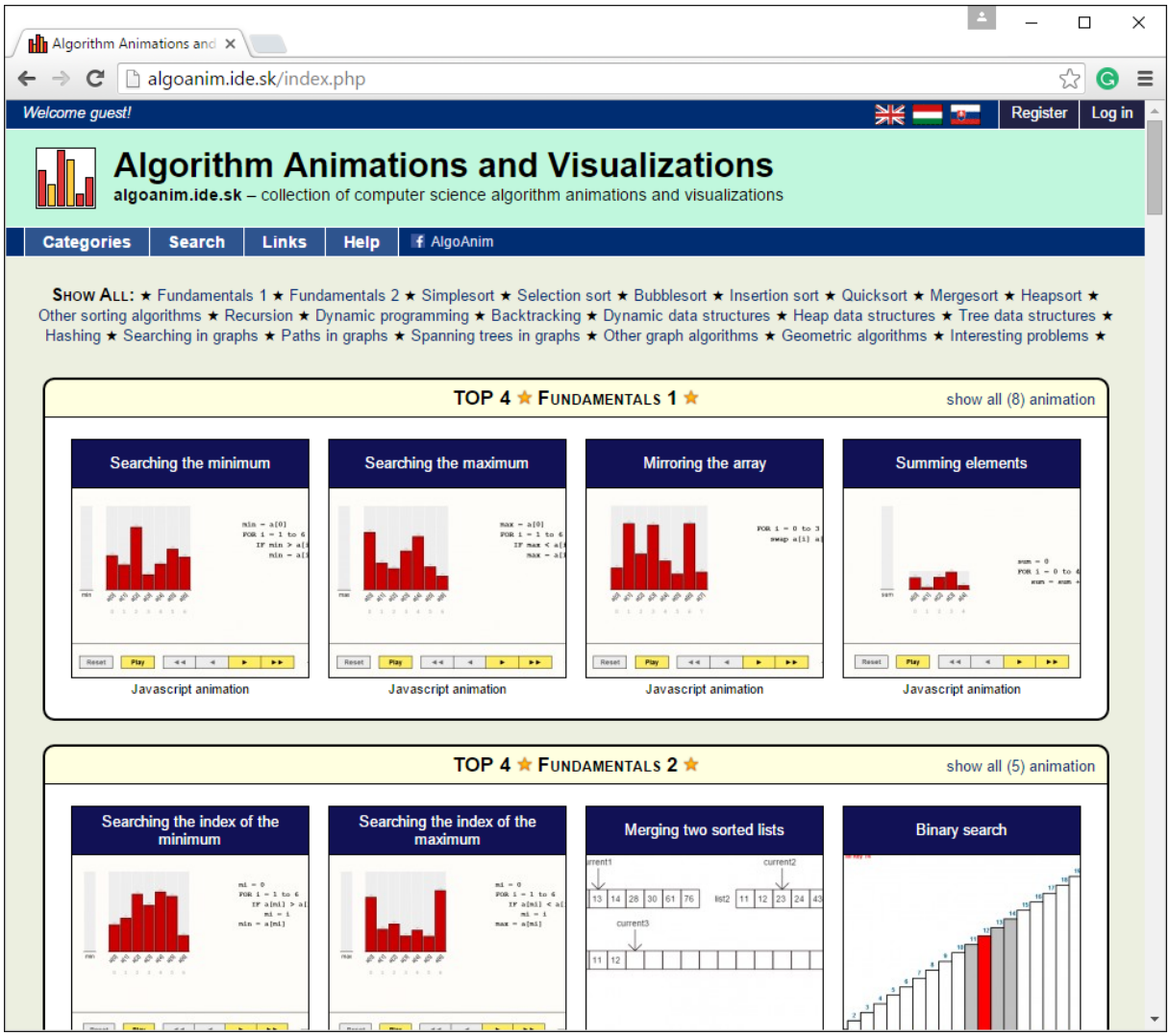
adatszerkezetek. Mindezek helyettesíthetők az LSL nyelvben megtalálható lista adatszerkezettel, azonban ezek használata sok esetben nehézkes és körülményes.

A programozás oktatása a virtuális világokban szorosan összefügg a 3D modellek elkészítésével és textúrázásával, hiszen a diákoknak előbb el kell készíteniük azt a virtuális világbeli objektumot, melyet interaktívvá szeretnének tenni a szkriptjeik segítségével. Egy megoldás lehet a programozás oktatás kezdeti szakaszában, ha a tanár előre elkészíti ezeket a szkript nélküli modelleket és textúrákat, majd megosztja őket diákokkal. Egy ilyen módszer bemutatása található a (Végh, 2014b) publikációban.

2. **Tanár vagy másik felhasználó által előre elkészített interaktív modellek használatával** – Elsősorban az algoritmusok oktathatók ilyen formában, ahogy azt az előző fejezetekben is bemutattuk. Az ilyen oktatási modellek elkészítése azonban sokszor bonyolult és jelenleg még eléggé korlátozott a virtuális világ adta technikai lehetőségekkel, használatuk pedig a diákok számára kisebb motiváló erővel jár, mint saját modellek létrehozása és azok interaktívvá tétele saját szkriptjeik segítségével.

5 Algoritmusok animációinak gyűjteménye

A doktori munkánkkal kapcsolatos fejlesztés következő szakaszában a célunk egy olyan internetes portál létrehozása és üzemeltetése volt, melyen összegyűjtöttünk különböző, weboldalba beágyazható algoritmus-animációkat. Ez a portál az http://algoanim.ide.sk/ címen érhető el (Végh, 2016a, 2016b).



25. ábra: Algoritmusok animációinak gyűjteménye

A weboldalt háromnyelvűre készítettük el: angol, magyar és szlovák. Mindhárom nyelv alatt olyan animációk találhatók, amelyekben az algoritmusok magyarázata az adott nyelven szerepel. Így tehát különböző animációk találhatók az oldal angol, magyar vagy szlovák részén. Jelenleg a portál összesen 131 animációt tartalmaz: 72 angol nyelvűt, 35 magyar nyelvűt és 24 szlovák nyelvűt.

Az animációk kiválogatásánál és a saját animációink elkészítésénél (melyeket részletesen a következő fejezetekben tárgyalunk) ügyeltünk a doktori értekezés 2. fejezetében felsorolt multimédia elvekre és interaktivitással kapcsolatos javaslatokra. Bízunk benne, hogy így sikerült egy olyan gyűjteményt létrehoznunk, amely sikeresen felhasználható az informatikai algoritmusok oktatásában.

5.1 A gyűjtemény animációinak besorolása kategóriákba különböző szempontok alapján

Az animációkat mindenekelőtt a szemléltetett algoritmusok alapján soroltuk be.

Kategóriák:		
Elemi algoritmusok 1	Egyéb rendezési algoritmusok	Keresés gráfokban
Elemi algoritmusok 2	Rekurzió	Utak a gráfokban
Egyszerű rendezés	Dinamikus programozás	Feszítőfák gráfokban
Kiválasztó rendezés	Backtracking	Egyéb gráfalgoritmusok
Buborékrendezés	Dinamikus adatszerkezetek	Geometriai algoritmusok
Beszúró rendezés	Kupac adatszerkezetek	Egyéb algoritmusok
Gyorsrendezés	Fa adatszerkezetek	Érdekes problémák
Összefésülő rendezés	Hashelés	
Kupacrendezés	Gráf reprezentálása	

4. táblázat: Az algoanim.ide.sk portál kategóriái

Mindegyik animációhoz továbbá hozzárendeltük az animáció elkészítéséhez felhasznált technológiát, az animáció interaktivitásának a fokát, és az animációban használt programozási nyelvet (amennyiben az animáció tartalmazott programkódot).

Technológia:	**Interaktivitás:**	**Programozási nyelv:**
JavaScript animáció	Nincs	C/C++
Videó	Alacsony	HTML
Animált kép	Közepes	Java
Java applet	Magas	JavaScript
Flash animáció		Pascal
Prezentáció		PHP
		Visual Basic
		Pszeudokód

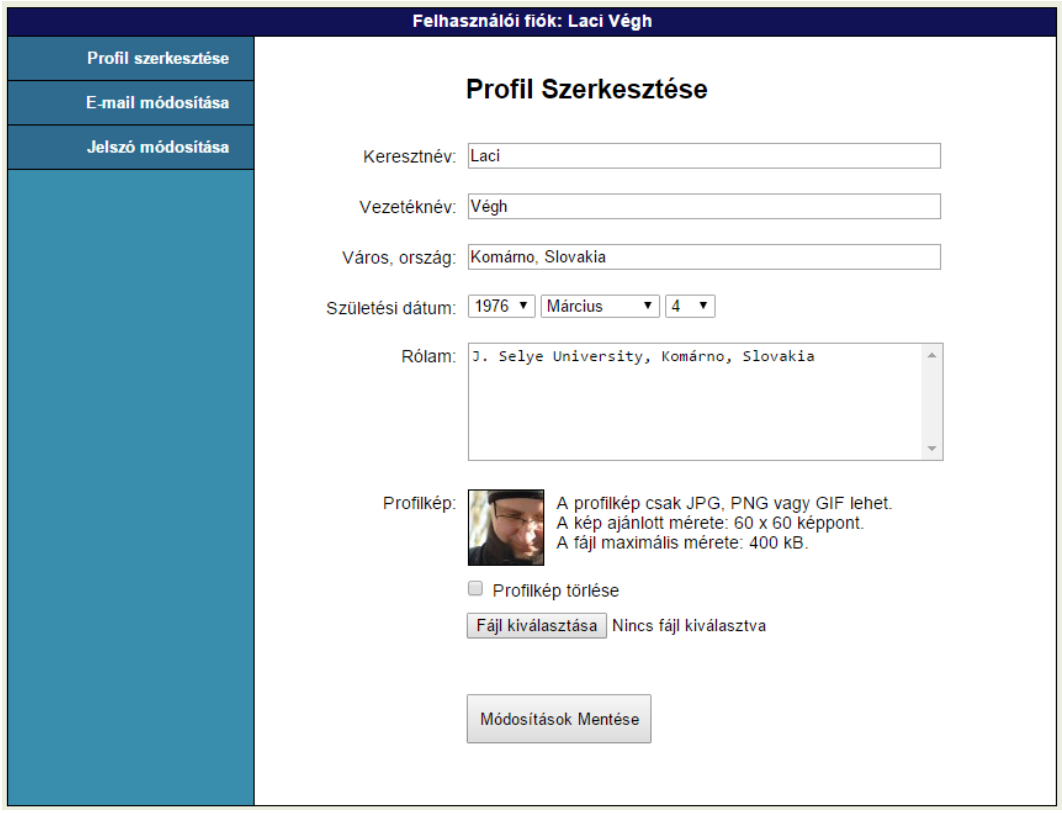
5. táblázat: Az animációk lehetséges besorolásai a felhasznált technológia, interaktivitás, és programozási nyelv alapján

Az interaktivitás fokának meghatározásakor az alábbi kategorizálásból indultunk ki:

- **Nincs interaktivitás** – a felhasználók nem tudják befolyásolni az animáció menetét, pl. ilyenek az animált gif képek,
- **Alacsony interaktivitás** - a felhasználók csupán elindíthatják, leállíthatják, esetleg léptethetik az animációt, pl. ilyenek a youtube videók,
- **Közepes interaktivitás** - a felhasználók saját bemeneti adatokat adhatnak meg, esetleg valamilyen módon kiválaszthatják a bemeneti adatok halmazát,
- **Magas interaktivitás** - a felhasználók kiválaszthatják, majd egérrel mozgathatják az animáció egyes részeit.

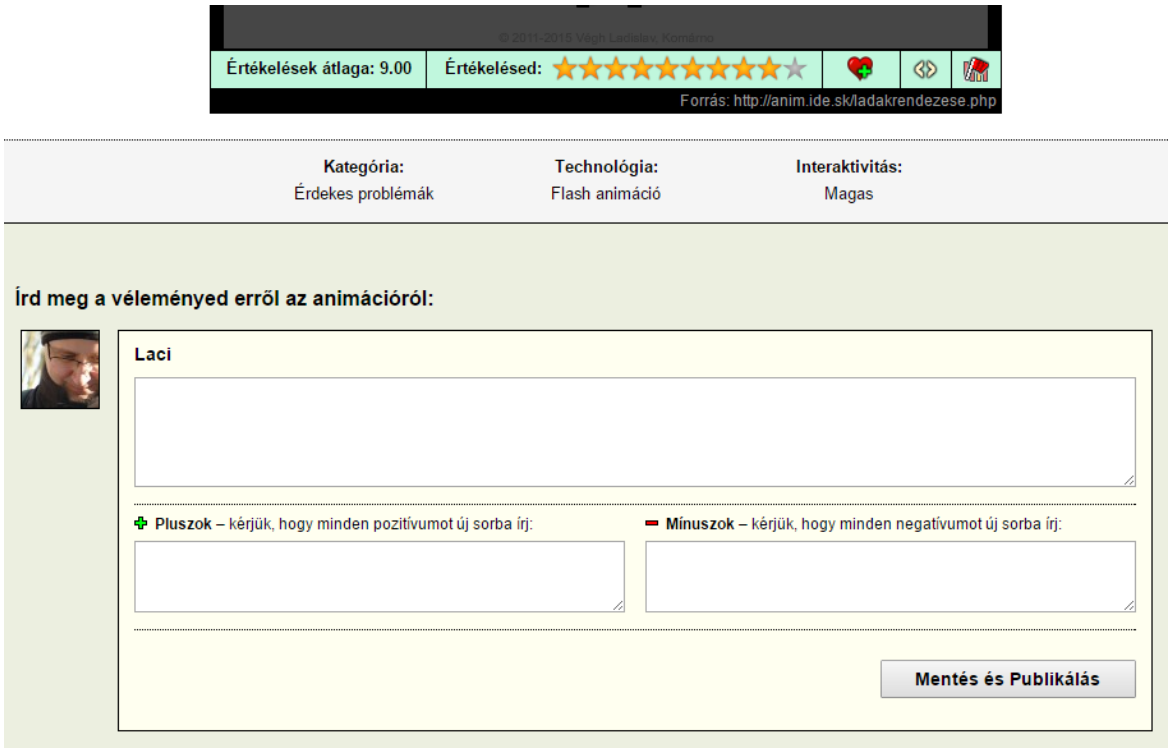
5.2 A weboldal felhasználói felülete

A weboldal egy egyszerű regisztrációs lehetőséget kínál a felhasználóknak, amely után további lehetőségek válnak elérhetővé. A regisztrált felhasználók természetesen utólag is bármikor módosíthatják a regisztráció során megadott adataikat.



26. ábra: Felhasználói profil, e-mail cím és jelszó módosítására szolgáló felület

Az oldal továbbá lehetőséget kínál pontszámmal (csillagokkal) értékelni az egyes animációkat (1-től 10-ig), szóbeli értékelést írni, felsorolni az animáció pozitívumait és negatívumait.



27. ábra: Animáció értékelése pontszámmal (csillagokkal) vagy vélemény írásával

A regisztrált felhasználók továbbá hozzáadhatják bármelyik animációt a kedvenceik közé, így ezeket később könnyedén elérhetik a főmenü „Kedvencek” menüpontja alatt.

Mindegyik animációnál lehetőséget kínálunk más oktatási weboldalakba, blogokba, elektronikus tananyagokba való beágyazásra szolgáló HTML kód kigenerálására is.

28. ábra: Animáció hozzáadása kedvencekhez, beágyazásra szolgáló HTML kód megjelenítésére és a nem működő animáció jelentésére szolgáló nyomógombok

Mivel a portálon több internetes forrásból is gyűjtöttünk össze animációkat, melyeket folyamatosan karban szeretnénk tartani a jövőben is, a felhasználóknak lehetőséget adtunk új animációk gyűjteménybe való besorolásának javaslására és a gyűjteményben található nem működő animációk jelentésére. Új animációk javaslásáról és a nem működő animációk jelentéséről az adminisztrátor e-mailben is kap értesítést.

29. ábra: Új animáció gyűjteménybe való besorolásának javaslására szolgáló űrlap a felhasználói felületen

5.3 Kiadói, szerkesztői, és moderátori jogosultságok

A weboldalon a felhasználókhoz különféle jogosultságok rendelhetők. Ezeket az adatbázis **user_groups** táblájában definiáltuk az alábbi módon:

id	name	permissions
1	Standard User	
2	Editor	{"editor": 1}
3	Publisher	{"editor": 1, "publisher": 1}
4	Moderator	{"moderator": 1}
5	Editor & Moderator	{"editor": 1, "moderator": 1}
6	Publisher & Moderator	{"editor": 1, "publisher": 1, "moderator": 1}

30. ábra: Az oldal felhasználóinak jogosultságait tartalmazó tábla

Minden újonnan regisztrált felhasználóhoz a **Standard user (id: 1)** csoport van automatikusan hozzárendelve, melyet csak az oldal adminisztrátora tud módosítani. Az ilyen standard felhasználóknak az előző fejezetben említett jogosultságaik vannak.

Az **Editor (id: 2)** az animációkhoz tartozó rövid jellemzések és használati utasítások szövegeit is szerkesztheti, azonban az animációkhoz tartozó egyéb beállítások módosítására nincs joga.

A **Publisher (id: 3)** az animációk összes beállítását módosíthatja, beleértve az animációkat megjelenítő HTML kódokat, hivatkozásokat az animációk forrásaira, az animációk

képernyőképeit, az animációk besorolását kategóriákba, a felhasznált technológia, interaktivitás, programozási nyelv hozzárendelését az animációkhoz. A kiadói (publisher) jogokkal rendelkező személy továbbá láthatóvá vagy láthatatlanná teheti bármelyik animációt az oldal felhasználói számára, a felhasználói javaslatok alapján új animációkkal bővítheti a gyűjteményt, vagy kitörölhet animációkat a gyűjteményből.

A **Moderátor (id: 4)** az animációkhoz írt felhasználói véleményeket moderálhatja (szerkesztheti, törölheti, elrejtheti a weboldalról), azonban az animációk bárminemű szerkesztésére nem jogosult.

A felhasználói csoportokat definiáló táblában kialakított további bejegyzések a már említettek jogosultságok kombinációi: **Editor & Moderator (id: 5)**, **Publisher & Moderator (id: 6)**.

5.4 A weboldal belső szerkezetének rövid jellemzése

A portált lokális szerveren (XAMPP 3.2.1), Eclipse Luna 4.4.1 fejlesztői környezetben hoztuk létre. A weboldal elkészítéséhez MySQL 5.6.11, PHP 5.5.9, HTML 5, CSS 3, JavaScript, Ajax technológiákat használtunk.

A portálhoz létrehozott adatbázis az alábbi táblákból áll, az adatbázis teljes szerkezete és a táblák közötti kapcsolat a munka 3. mellékletében található.

Az animációkkal kapcsolatos táblák:	A felhasználókkal kapcsolatos táblák:
anims anims_broken anims_category anims_category_tags anims_proglang anims_progland_tags anims_technology anims_technology_tags anims_ratings anims_reviews anims_favorites anims_suggested anims_suggested_category anims_suggested_proglang anims_suggested_technology	users users_groups users_verifications users_sessions users_emailchange users_lostpasswords

6. táblázat: A MySQL adatbázisban kialakított táblák listája

A webszerver tárhelyén kialakított mappák szerkezete a 4. mellékletben található.

A tárhely főmappájában helyezkedik el a kezdőoldal állománya, az e-mail cím hitelesítésére és az elveszett jelszó helyreállítására szolgáló állományok, a webszerveren bizonyos időintervallumonként lefutó CRON állomány (adatbázis esetleges tisztítását végzi), és néhány PHP állomány az AJAX technológiák használatához.

5.5 Jövőbeli tervek, továbbfejlesztési lehetőségek

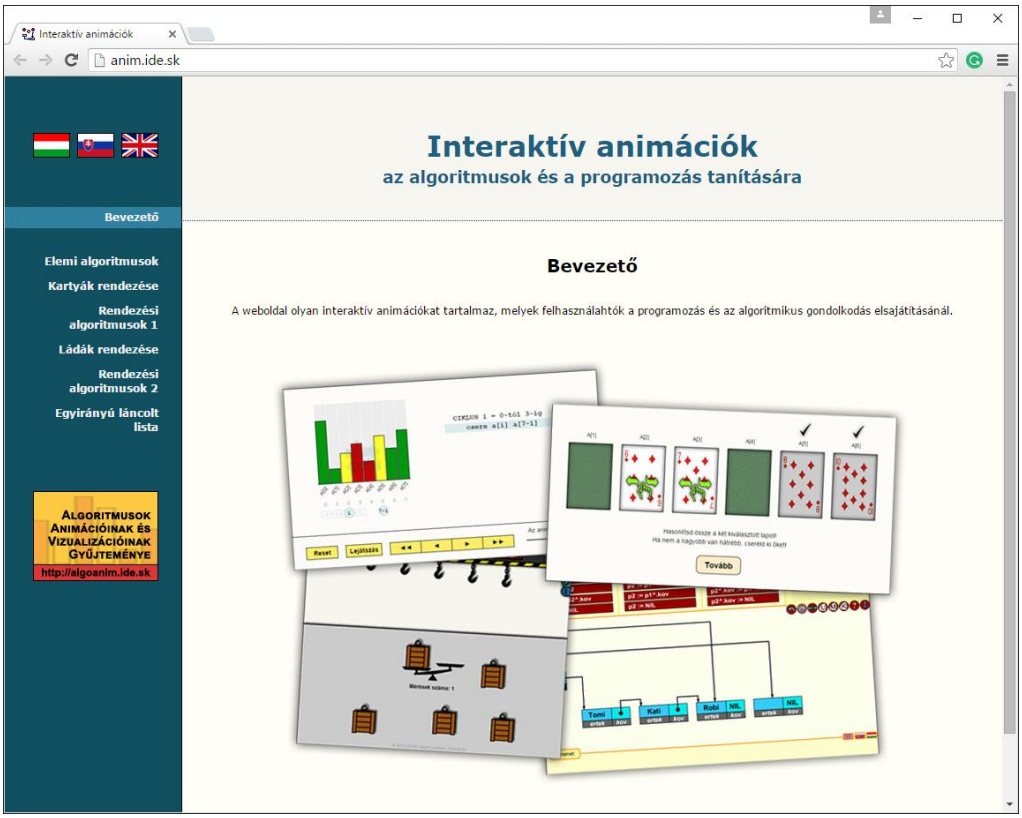
A jövőben szeretnénk a weboldalt folyamatosan bővíteni algoritmusok oktatásnál sikeresen felhasználható animációkkal és vizualizációkkal. Célunk egy olyan átfogó portál létrehozása, amely segítségével az informatika szakos hallgatók könnyebben és gyorsabban megérthetik bármilyen informatikai algoritmus működését. A hallgatókat arra buzdítjuk, hogy értékeljék az itt található animációkat, így egy bizonyos idő után kiszelektálódhatnak a tanulmányaik során sikeresebben felhasználhatók.

Az újabb animációk hozzáadásakor a már létező gyűjteményhez továbbra is ügyelünk arra, hogy azok a diákok számára könnyen érthetők legyenek, ill. minél jobban megfeleljenek a doktori értekezés 2. fejezetében felsorolt elveknek és javaslatoknak.

6 Weboldalba beágyazható saját fejlesztésű interaktív animációk

A doktori kutatás következő részében saját, weboldalba beágyazható interaktív animációkat készítettünk elsős informatika szakos hallgatók részére, majd ezen animációk oktatásban való felhasználásukkal kapcsolatos pedagógiai kutatásokat végeztünk. Mindegyik animáció megtervezésénél próbáltuk az áttekintett irodalmi források eredményeit figyelembe venni és az animációkat ezek alapján megtervezni.

Az általunk készített interaktív animációkat besoroltuk az előző fejezetben említett gyűjteménybe, azonban az animációk az http://anim.ide.sk weboldalon is elérhetők három nyelven (magyar, szlovák, angol).



31. ábra: A létrehozott interaktív animációkat tartalmazó weboldal kezdőoldala

Az animációkat hat témakörbe soroltuk be olyan sorrendben, ahogy a diákok a programozás órán megismerkednek velük:

- **Elemi algoritmusok** (részletesen a 10. fejezetben tárgyaljuk) – két változó cseréjét, tömb elemeinek összegszámítását, tömb tükrözését, maximum és minimum keresését, maximum és minimum indexének keresését bemutató animációk.

- **Kártyák rendezése** (részletesen a 7. fejezetben tárgyaljuk) – egyszerű cserés rendezés, buborékrendezés, beszúró rendezés, minimum- és maximumkiválasztásos rendezés algoritmusainak lényeges lépéseit bemutató interaktív animációk. Az animációk nem térnek ki az egyes algoritmusok részletes bemutatására, helyette játékos formában – kártyalapok rendezésével – ismertetik meg a hallgatókat a rendezési algoritmusok lényeges lépéseivel és az algoritmusok közti különbségekkel.
- **Rendezési algoritmusok 1** (részletesen a 10. fejezetben tárgyaljuk) – egyszerű cserés rendezés, buborékrendezés, továbbfejlesztett buborékrendezés, beszúró rendezés, továbbfejlesztett beszúró rendezés, minimum- és maximumkiválasztásos rendezés algoritmusainak részletes bemutatására szolgáló mikro-szintű interaktív animációk.
- **Ládák rendezése** (részletesen a 8. fejezetben tárgyaljuk) – az animáció játékos formába – ládák rendezésével – ismerteti meg a diákokat a rendezési algoritmusok fogalmával. A játék egyik célja, hogy a rendezési algoritmusokat még nem ismerő diákok „kifejlesszék" saját rendezési algoritmusukat. Egy másik cél, hogy a diákok az általuk megalkotott algoritmusban minimalizálják a mérések (összehasonlítások) számát. Az animáció hasznos oktatási segédeszközként szolgálhat a gyorsrendező algoritmus lényeges lépéseinek bemutatására is.
- **Rendezési algoritmusok 2** (részletesen a 10. fejezetben tárgyaljuk) – gyors- és összefésülő rendezési algoritmusokat bemutató mikro-szintű interaktív animációk.
- **Egyirányú láncolt lista** (részletesen a 9. fejezetben tárgyaljuk) – Pascal nyelv utasításai segítségével a diákok kialakíthatnak egyszerű láncolt listákat, így könnyebben megérthetik a mutatók használatát. Az animáció során folyamatosan látják a memóriában kialakuló listaelemek szemléltetését és a mutatókat reprezentáló nyilakat.

A munka további fejezeteiben részletesen bemutatjuk az itt felsorolt animációkat és az azokkal kapcsolatos pedagógiai kísérleteket.

Előbb a játék alapú, az algoritmusok működésének lényeges eseményeit bemutató animációkkal foglalkozunk (kártyák rendezése, ládák rendezése), majd az egyirányú láncolt lista kialakításának elsajátítására megalkotott animációt mutatjuk be.

Ezek után fogjuk tárgyalni a mikro-szintű animációk létrehozását elősegítő „inalan” JavaScript könyvtárat, majd a könyvtár segítségével létrehozott animációkat (elemi algoritmusok, rendezési algoritmusok).

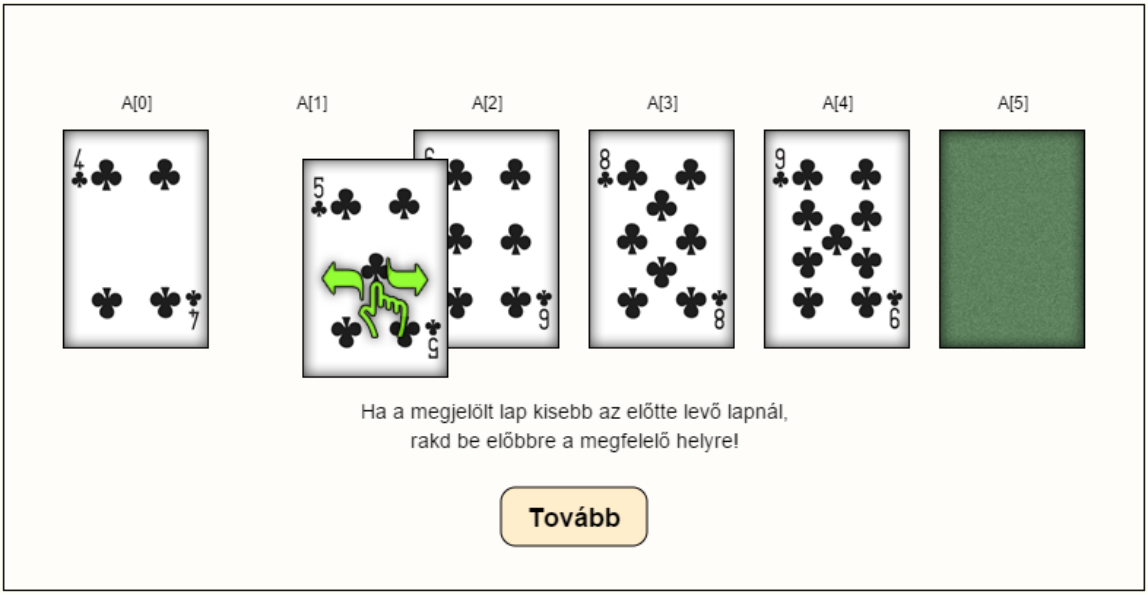
7 Interaktív rendezés kártyalapok segítségével

Ahogy azt már a 2.4 fejezetben említettük, Hansen és társai (Hansen et al., 2002) több olyan animáció egymás utáni használatát javasolják az oktatásban, mely ugyanazon algoritmust különbözőképpen mutatja be. Az általuk javasolt első típusú animációk az algoritmusok működésének lényeges eseményeit szemléltetik konkrét objektumok segítségével (pl. kártyalapok, ládák, különböző magasságú fenyőfák), nem mennek bele az absztrakt fogalmakba és az algoritmusok részleteibe.

Az itt bemutatásra kerülő animációk is ilyen típusúak. Az animációkban kártyalapok segítségével próbáljuk meg megismertetni a diákokat néhány rendezési algoritmus működésével és az azok közötti lényeges különbségekkel. A gyűjtemény öt rendezési algoritmus interaktív animációját tartalmazza három nyelven (magyar, szlovák, angol):

- egyszerű cserés rendezés,
- buborékrendezés,
- beszúró rendezés,
- minimumkiválasztásos rendezés,
- maximumkiválasztásos rendezés.

Ezek a következő weboldalon érhetők el: http://anim.ide.sk/kartyarendezes.php, de felkerültek az http://algoanim.ide.sk portálra is, ahol az elektronikus tananyagokba való beágyazásukra felhasználható HTML kódok is megjeleníthetők.



32. ábra: A beszúró rendezési algoritmus interaktív animációja

7.1 A szoftver felhasználói felületének bemutatása

Az animációk kialakításánál egyik fő célunk az volt, hogy azok használata minél interaktívabb legyen. Az általunk kialakított animációknál a felhasználók nem csupán passzív megfigyelői az animációknak, a rendezési algoritmusokba aktívan be kell kapcsolódniuk.

Mindegyik rendezésnél magát az animálást a felhasználók végzik bizonyos szabályok betartásával. Ezen szabályok a kártyalapok mozgatását az adott algoritmus egyes lépéseire korlátozzák. Így a felhasználók játékos módon, a játékszabályok folyamatos megismerésével sajátítják el az egyes rendezési algoritmusok lényeges lépéseit.

A diákok az adott algoritmus következő lépésére csak abban az esetben tudnak továbbmenni, ha az aktuális lépést már sikeresen végrehajtották.

7.2 A szoftver belső szerkezetének rövid jellemzése

Az animációk elkészítéséhez a HTML5 Canvas elemét, a JavaScript nyelvet és a CreateJS könyvtárat (http://www.createjs.com/) használtuk. Azért e technológiák alkalmazása mellett döntöttünk, mivel a HTML5 megjelenése óta (2014. október 28.) a weboldalon megjelenítendő animációk készítésére ezek kezdenek standarddá válni, kiszorítva más megoldásokat (pl. Adobe Flash animációkat). Nagy előnyük, hogy a felhasználóknak nem kell külön beépülő modult (plugin) használniuk az animációk megjelenítésére, hiszen a HTML5 nyelvet mindegyik jelenleg használt internet böngésző támogatja.

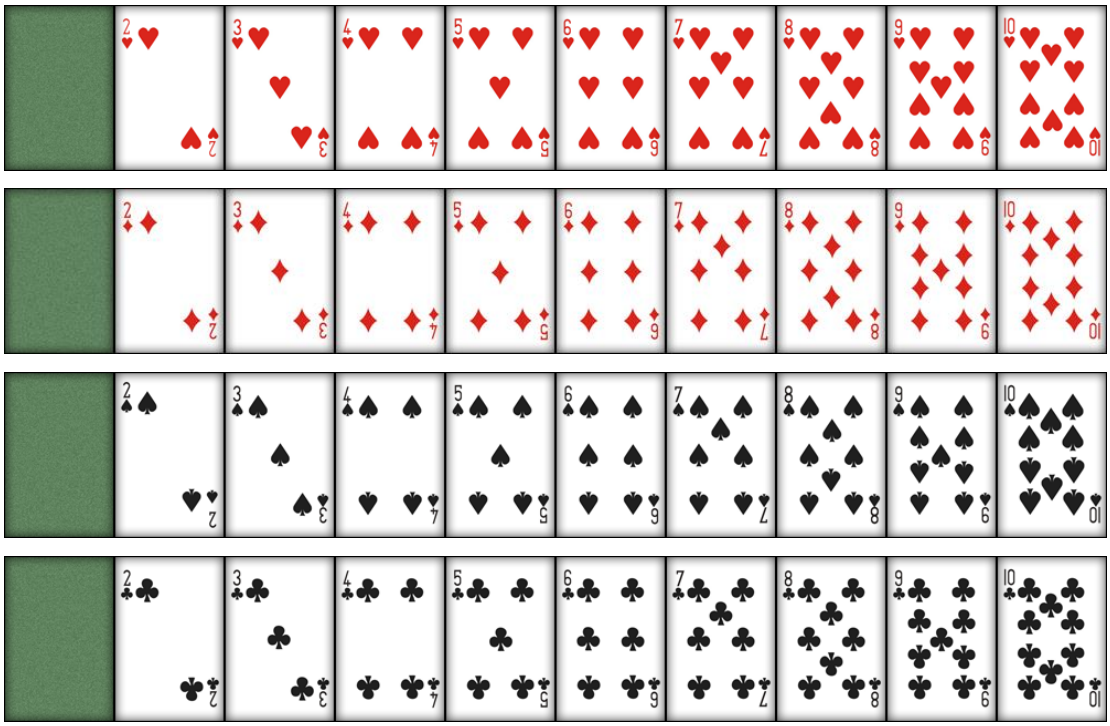
Az animációk elkészítéséhez létrehoztunk három osztályt, melyeket a CreateJS.Container osztályából származtattunk:

- **Button (button.js)** – az animációkban a „Tovább” nyomógomb és a nyomógomb fölötti magyarázat létrehozására használt osztály.
- **Card (card.js)** – egy kártyalap létrehozására használt osztály.
- **Cards (cards.js)** – több kártyalap létrehozására használt osztály.

Minden egyes rendezési algoritmus animációjához külön JavaScript állomány tartozik, amely a HTML5 Canvas elemének azonosítója (id) alapján határozza meg, hogy a weboldal melyik részén legyen az adott animáció megjelenítve. Ezen JavaScript állományok: **simplesort.js**, **bubblesort.js**, **insertsort.js**, **minsort.js**, **maxsort.js**.

Ahhoz, hogy a diákok minél könnyebben megértsék a szemléltetett rendezési algoritmusokat, az összes megjelenített kártyalap egy adott animáción belül mindig ugyanolyan

színű. Az egyes animációkban használt szín véletlenszerűen van kiválasztva a rendelkezésre álló négy színből az inicializálásakor. Az animációk jobb érthetősége végett a kártyalapok csak a 2 – 10 lapok közül vannak kiválasztva (a J, Q, K, A lapok nincsenek használva az animációkban).



33. ábra: Az animációkban használt kártyalapokat tartalmazó képek

7.3 Kísérlet bemutatása

Az előző alfejezetben bemutatott, kártyalapokat használó interaktív animációk segítségével pedagógiai kísérletet végeztünk a Selye János Egyetemen a 2014/15-ös és 2015/16-os akadémiai év nyári szemeszterében az „Algoritmizáció és programozás” óra keretén belül. A kísérletben 2014/15-ben 39, 2015/16-ban 53 elsős informatika szakos hallgató vett részt. A célunk annak felmérése volt, hogy a diákok képesek-e az interaktív animációk segítségével felismerni az egyes rendezési algoritmusok jellegzetes vonásait és az azok közti különbségeket (Végh, 2016e; Végh & Stoffová, 2017; Végh & Takáč, 2017).

A diákok többsége már foglalkozott valamilyen rendezési algoritmussal a kísérlet előtt is, azonban néhány diák nem tanult korábban semmit sem a rendezési algoritmusokról. A hallgatók rengeteg feladatot oldottak meg tömbökön, pl. legkisebb vagy legnagyobb elem keresése, bizonyos elemek megszámlálása, elemek összeadása, átlagszámítás, tömb tükrözése. Ezeken kívül képesek voltak pszeudokódokat is értelmezni.

A kutatást kérdőívek (tesztek) segítségével valósítottuk meg. Az első kérdőívet a hallgatók az animációkkal való kísérletezés előtt töltötték ki (előzetes tudás felmérése céljából), a második kérdőívet az animációkkal való kísérletezés után. Mindkét kérdőív az alábbi táblázatot tartalmazta (az oszlopok és a sorok A1–E7-tel való megjelölése nélkül), amely kitöltése során a diákok feladata az volt, hogy jelöljék meg X-el mindegyik állítás (1-7) sorában azokat a rendezési algoritmusokat (A-E), melyekre az adott állítás igaz.

		A	**B**	**C**	**D**	**E**
		Egyszerű cserés rendezés (Simplesort)	**Buborékrendezés** (Bubblesort)	**Beszúró rendezés** (Insertion sort)	**Minimumkiválasztásos rendezés** (Selection sort: Minsort)	**Maximumkiválasztásos rendezés** (Selection sort: Maxsort)
1	Mindig egymás melletti két szomszédos elemet hasonlítunk össze.					
2	Mindegyik elemet összehasonlítjuk az összes mögötte levővel.					
3	Mindig előbb végignézzük a rendezetlen elemeket, melyek közül kiválasztunk egyet, majd ezt az elemet cseréljük fel a tömbrész elején vagy végén levő elemmel.					
4	Minden egyes végigfutásnál a **legkisebb elem** a rendezetlen rész **elejére** kerül.					
5	Minden egyes végigfutásnál a **legnagyobb elem** a rendezetlen rész **végére** kerül.					
6	A rendezés alatt, a tömb elején vagy végén kialakult rendezett részben az elemek **már nem módosulnak (nem tolódnak arrébb)** a rendezés befejezéséig.					
7	A rendezés alatt, a tömb elején vagy végén kialakult rendezett részben az elemek **még módosulhatnak (arrébb tolódhatnak)** a rendezés befejezéséig.					

7. táblázat: A rendezési algoritmusok közötti különbségek megértését felmérő táblázat (2015/16)

A kérdőívek további részében az algoritmusok neveit kellett a pszeudokódjaikkal összekötniük. A második kérdőívben ezeken kívül a diákok animációkkal kapcsolatos véleményeire is kíváncsiak voltunk.

A doktori értekezés 5. – 8. mellékletében megtalálhatók a 2014/15-ben használt tesztek, ill. a 2015/16-ban kissé pontosított tesztek (a felmérésben szereplő táblázat 4. és 5. állítása, ill. a minimumkiválasztásos és a maximumkiválasztásos rendezés pszeudokódja volt pontosítva).

Mivel a diákok közül többen is voltak, akik nem foglalkoztak semmilyen rendezési algoritmussal a kísérlet előtt, annak elkerülése érdekében, hogy a diákok ne tippeljenek, megkértük őket, hogy csak az általuk ismert rendezési algoritmusokkal kapcsolatos részeket töltsék ki a kísérlet alatt. Az elő- és utótesztelés során tehát akár teljesen üres lapot is leadhattak. A meggondolatlan tippelés ellen az is motiválta őket, hogy a tesztekben megjelölt helyes válaszokra +1, helyetlen válaszokra -1 pontot kaptak, melyek beszámítottak a félévi gyakorlati értékeléseikbe is.

7.4 Eredmények kiértékelése és elemzése

A diákok által adott válaszokat a 9. és a 10. melléklet tartalmazza. A kísérlet után megszámoltuk a megjelölt algoritmus-állítás kombinációkat az elő- és az utótesztelésben. Az alábbi táblázatokban láthatók a két teszt közötti különbségek, tehát hogy az utótesztelés során mennyivel több (+) vagy kevesebb (-) jelölést kaptak az egyes algoritmus-állítás kombinációk, mint az előtesztelés alkalmával.

	A	**B**	**C**	**D**	**E**
1. állítás:	-14	+12 **X**	+1 **?**	-2	-2
2. állítás:	+7 **X**	-6	+11	-6	-4
3. állítás:	+6	-6	-8	+21 **X**	+19 **X**
4. állítás:	+29 **X**	-2	+12 **?**	+3 **X**	0
5. állítás:	-1	+26 **X**	+2	+1	+2 **X**
6. állítás:	+18 **X**	+26 **X**	-2	+16 **X**	+15 **X**
7. állítás:	-10	-14	+14 **X**	-2	-2

8. táblázat: Az algoritmus-állítás kombinációk megjelöléseinek utó- és előtesztelés közötti különbségei (2014/15-ös akadémiai év)

	A	B	C	D	E
1. állítás:	-32	+6 **X**	-5 **?**	-3	-2
2. állítás:	+21 **X**	-6	+3	-1	-6
3. állítás:	+8	+1	+3	0 **X**	-4 **X**
4. állítás:	+11 **X**	-2	+14 **?**	+5 **X**	-1
5. állítás:	+5	+22 **X**	+2	-5	-1 **X**
6. állítás:	+15 **X**	+7 **X**	-14	+18 **X**	+17 **X**
7. állítás:	-3	-4	+22 **X**	-9	-6

9. táblázat: Az algoritmus-állítás kombinációk megjelöléseinek utó- és előtesztelés közötti különbségei (2015/16-os akadémiai év)

	A	B	C	D	E
1. állítás:	-46	+18 **X**	-4 **?**	-5	-4
2. állítás:	+28 **X**	-12	+14	-7	-10
3. állítás:	+14	-5	-5	+21 **X**	+15 **X**
4. állítás:	+40 **X**	-4	+26 **?**	+8 **X**	-1
5. állítás:	+4	+48 **X**	+4	-4	+1 **X**
6. állítás:	+33 **X**	+33 **X**	-16	+34 **X**	+32 **X**
7. állítás:	-13	-18	+36 **X**	-11	-8

10. táblázat: Az algoritmus-állítás kombinációk megjelöléseinek utó- és előtesztelés közötti különbségei (2014/15 és 2015/16 akadémiai év egyesítve)

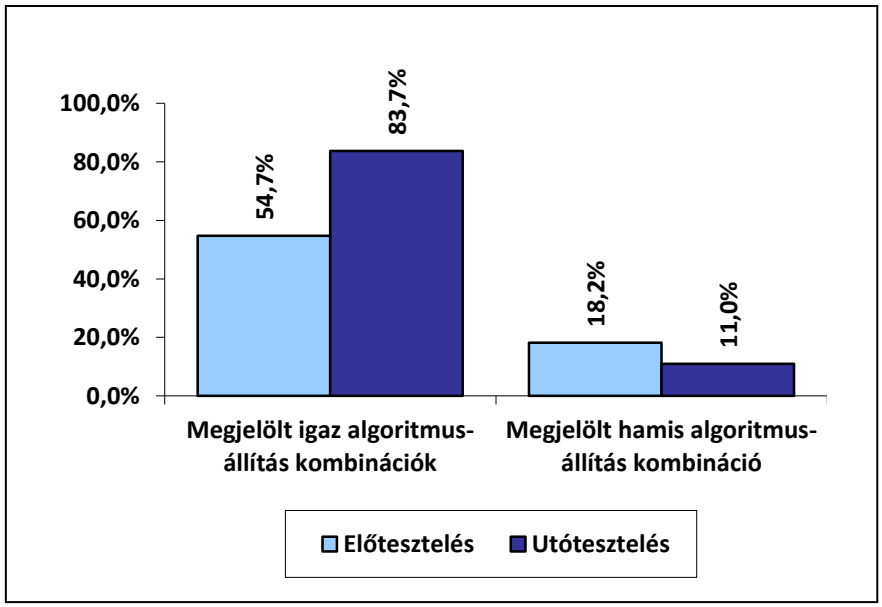
A táblázatokban X-szel jelöltük meg a helyes válaszokat. A 2014/15-ös év eredményeinek összeszámlálása után figyeltünk fel a C1, C4 mezőkre, melyeknél utólag derült ki, hogy az állítások nem teljesen egyértelműek. A táblázatokban ezen esetekben kérdőjelet tettünk a szám után. A kérdéses algoritmus-állítás kombinációk a következők voltak:

- C1 a beszúró rendezésre vonatkozik: „Mindig egymás melletti két szomszédos elemet hasonlítunk össze”. Ez az állítás igaz az egyszerű beszúró rendezésre, azonban hamis a továbbfejlesztett beszúró rendezésre. Mivel a beszúró rendezést szemléltető animáció csupán az algoritmus lényeges tulajdonságaira fókuszál, ezért nem derül ki belőle, hogy melyik beszúró rendezésről van szó. A diákok a rendezés során gondolhatnak bármelyikre a kettő közül.
- C4 szintén a beszúró rendezésre vonatkozik: „Minden egyes végigfutásnál a legkisebb elem a rendezetlen rész elejére kerül (a rendezett sorrend a tömb elejétől kezd kialakulni)”. A mondat első része nem igaz a beszúró rendezésre, hiszen a

legkisebb elem nem a rendezetlen rész elejére, hanem a rendezetlen rész előtti rendezett részbe kerül. A mondat második része viszont azt sugallja, hogy az állítás igaz a beszúró rendezésre, hiszen a rendezett sorrend a tömb elejétől kezd kialakulni. A 2015/16-ös teszteléshez használt kérdőívben ezt megpróbáltuk pontosítani a zárójelbe írt rész elhagyásával. Sajnos azonban így sem volt az állításunk egyértelmű. A diákok „rendezetlen résznek” az egész tömböt tekintették, mivel a beszúró rendezés alatt az egész tömb még folyamatosan módosul.

Mivel e két algoritmus-állítás kombináció X-szel való megjelölése és a nem megjelölése is elfogadható helyes válasznak, a kiértékelés további részében a C1 és a C4 mezőket nem vettük figyelembe.

Az alábbi grafikon szemlélteti, hogy a diákok az összes igaz algoritmus-állítás kombináció közül mennyit jelöltek meg (helyesen), ill. az összes hamis algoritmus-állítás kombináció közül mennyit jelöltek meg (helytelenül) az előtesztelés és az utótesztelés során.



34. ábra: A helyesen megjelölt igaz algoritmus-állítás kombinációk és a helytelenül megjelölt hamis algoritmus-állítás kombinációk

A helyesen megjelölt igaz algoritmus-állítás kombinációk száma 53,1%-al növekedett 654-ről (54,7%-ról) 1001-re (83,7%-ra). A növekedés részben várható volt, hiszen a diákok egy része korábban nem tanulta a rendezési algoritmusokat. Ebből azonban az is kiderült, hogy a diákoknak sikerült az animációk segítségével megérteniük a rendezési algoritmusok lényeges lépéseit és az azok közti különbségeket.

Továbbá megfigyelhetjük, hogy bár minden tesztelés során felhívtuk a diákok figyelmét arra, hogy csak azokat az algoritmus-állítás kombinációkat jelöljék meg, amelyeket tanultak és tudnak, mégis az előtesztelés során 335 (18,2%) hamis algoritmus-állítás kombinációt helytelenül megjelöltek. Az utótesztelésnél a helytelenül megjelölt hamis algoritmus-állítás kombinációk száma 202-re (11,0%-ra) csökkent, ami 39,7% csökkenést jelent.

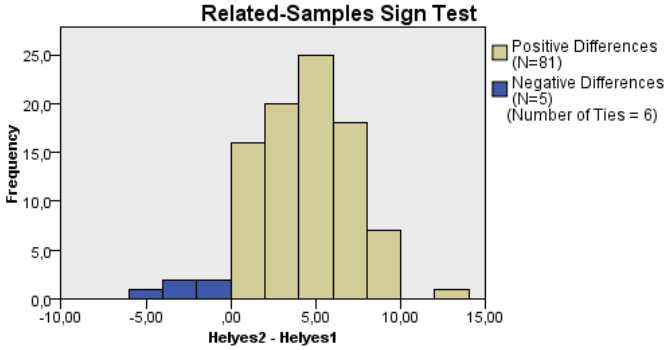
Annak érdekében, hogy megállapíthassuk, hogy a helyes válaszok növekedése és a helytelen válaszok csökkenése szignifikáns-e, a diákok által megjelölt algoritmus-állítás kombinációk számán páros előjelpróbát (paired sign test, SPSS Statistics, 2013) végeztünk. A teszteléshez azért nem a páros t-próbát vagy a Wilcoxon-féle előjeles rang-próbát választottuk, mivel az eloszlás se nem normál, se nem szimmetrikus. Az alábbi táblázatokban a Helyes1, Helyes2 jelölik a diákok által helyesen bejelölt algoritmus-állítás kombinációk számát, míg a Helytelen1, Helytelen2 jelölik a diákok által helytelenül bejelölt algoritmus-állítás kombinációk számát. A Helyes1, Helytelen1 az előtesztelésre, a Helyes2, Helytelen2 pedig az utótesztelésre vonatkozik.

Report

Median

Helyes1	Helyes2	helyes_diff
7,0000	11,0000	4,0000

Related-Samples Sign Test



Total N	92
Test Statistic	81,000
Standard Error	4,637
Standardized Test Statistic	8,087
Asymptotic Sig. (2-sided test)	,000

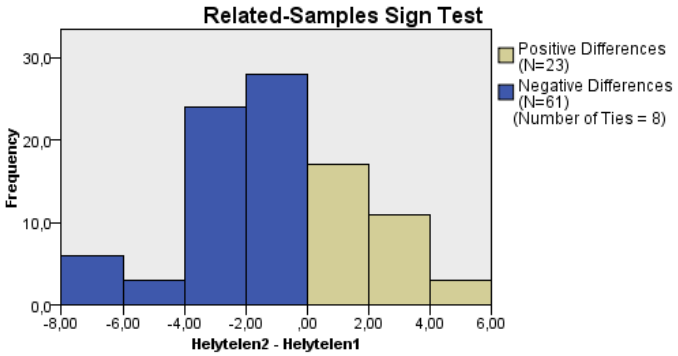
11. táblázat: Az előtesztelés (Helyes1) és utótesztelés (Helyes2) alatt az egyes diákok által megjelölt algoritmus-állítás kombinációk számán végzett előjelpróba eredménye (SPSS Statistics, 2013)

A diákoknak az animációk használatával sikerült szignifikánsan több helyes algoritmus-állítás kombinációt megjelölniük a táblázatban: 81 diák jelölt be több helyes választ, 5 diák jelölt be kevesebb helyes választ, és 6 diáknál nem látható változás. Az elő- (medián: 7) és utótesztelés (medián: 11) során bejelölt helyes válaszok száma közti különbségek mediánja **4** válasz, **z = 8.087, p < 0.0005**.

Report

Median

Helytelen1	Helytelen2	helyetelen_diff
4,0000	2,0000	-1,0000



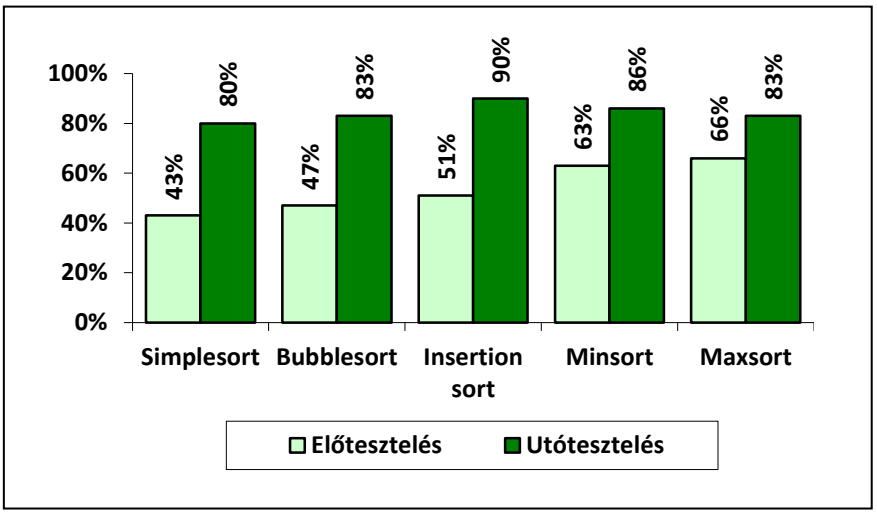
Total N	92
Test Statistic	23,000
Standard Error	4,583
Standardized Test Statistic	-4,037
Asymptotic Sig. (2-sided test)	,000

12. táblázat: Az előtesztelés (Helytelen1) és utótesztelés (Helytelen2) alatt az egyes diákok által megjelölt algoritmus-állítás kombinációk számán végzett előjelpróba eredménye (SPSS Statistics, 2013)

A helytelenül megjelölt algoritmus-állítás kombinációk száma is szignifikánsan csökkent az animációk használatával: 61 diák jelölt be kevesebb helytelen választ, 23 diák jelölt be több helytelen választ és 8 diáknál nem látható változás. Az elő- (medián: 4) és utótesztelés (medián: 2) során bejelölt helytelen válaszok száma közti különbségek mediánja **-1** válasz, **z = -4.037, p < 0.0005**.

Azt is megvizsgáltuk, hogy melyik rendezési algoritmusoknál adtak meg a diákok helyes vagy helytelen válaszokat. Ezekből kiderülhet, hogy az öt interaktív animáció közül melyik volt a diákok számára a legkönnyebben vagy a legnehezebben érthető.

A rendezési algoritmusok szerint csoportosított, helyes megjelölések számának arányát az alábbi grafikon szemlélteti.



35. ábra: A megjelölt igaz algoritmus-állítás kombinációk aránya (helyes válaszok) az egyes rendezési algoritmusok szerint csoportosítva

Annak érdekében, hogy megállapíthassuk, hogy a helyes válaszok növekedése szignifikáns-e, mindegyik rendezési algoritmusra külön-külön elvégeztük az előjelpróbát a diákok által megjelölt igaz állítások számán.

Report

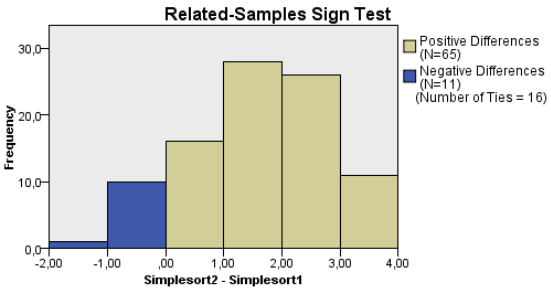
	Simplesort1	Simplesort2	Simplesort_diff
Median	1,00	3,00	1,000

	Bubblesort1	Bubblesort2	Bubblesort_diff
Median	1,00	3,00	1,000

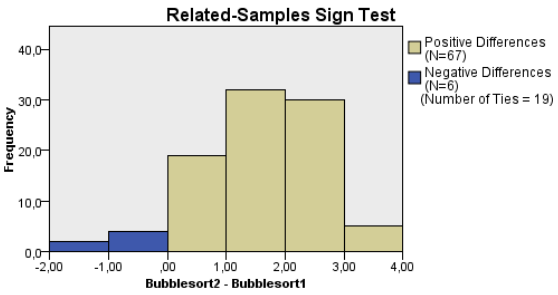
	Insertsort1	Insertsort2	Insertsort_diff
Median	1,00	1,00	0,000

	Minsort1	Minsort2	Minsort_diff
Median	2,00	3,00	1,000

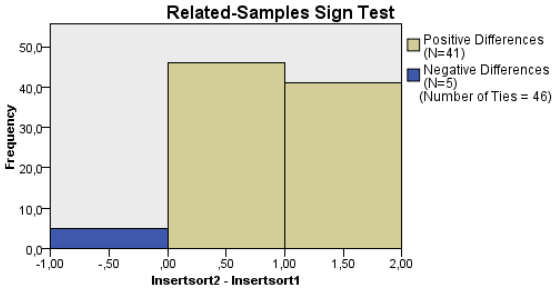
	Maxsort1	Maxsort2	Maxsort_diff
Median	2,00	3,00	0,000



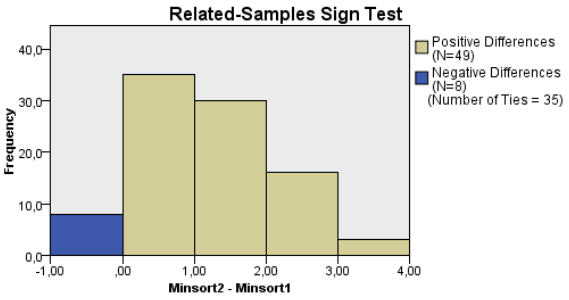
Total N	92
Test Statistic	65,000
Standard Error	4,359
Standardized Test Statistic	6,080
Asymptotic Sig. (2-sided test)	,000



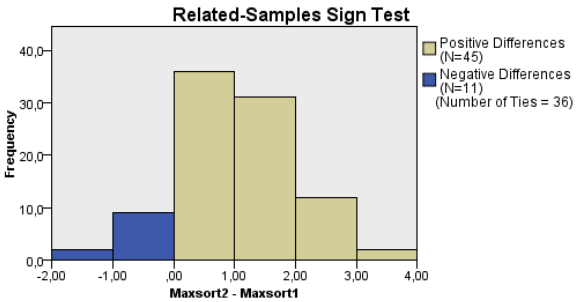
Total N	92
Test Statistic	67,000
Standard Error	4,272
Standardized Test Statistic	7,022
Asymptotic Sig. (2-sided test)	,000



Total N	92
Test Statistic	41,000
Standard Error	3,391
Standardized Test Statistic	5,160
Asymptotic Sig. (2-sided test)	,000



Total N	92
Test Statistic	49,000
Standard Error	3,775
Standardized Test Statistic	5,298
Asymptotic Sig. (2-sided test)	,000

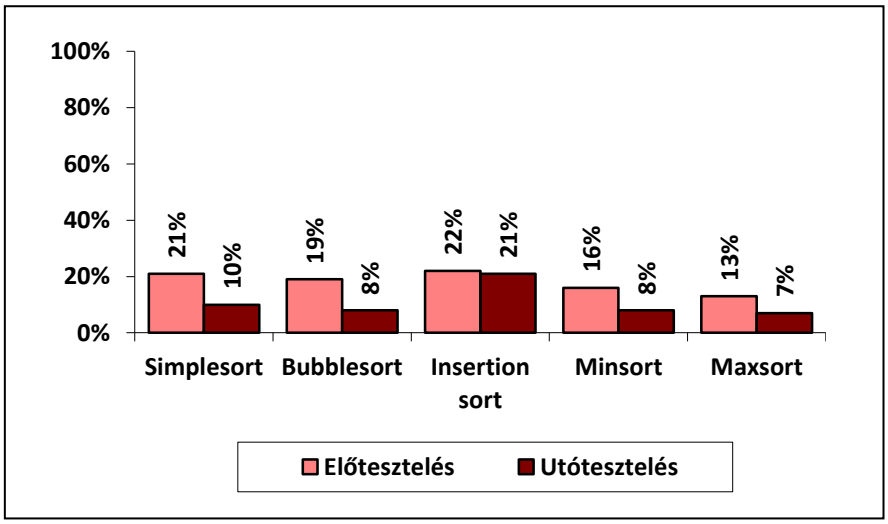


Total N	92
Test Statistic	45,000
Standard Error	3,742
Standardized Test Statistic	4,410
Asymptotic Sig. (2-sided test)	,000

13. táblázat: Az előtesztelés (Simplesort1, Bubblesort1, Insertsort1, Minsort1, Maxsort1) és utótesztelés (Simplesort2, Bubblesort2, Insertsort2, Minsort2, Maxsort2) alatt az egyes diákok által az algoritmusokhoz helyesen megjelölt állítások számán végzett előjelpróbák eredménye (SPSS Statistics, 2013)

A táblázatokból látható, hogy mindegyik rendezési algoritmusnál az előjelpróba szignifikáns eredményt mutatott, tehát a diákok szignifikánsan több helyes választ jelöltek meg mindegyik rendezési algoritmusnál az animációkkal való kísérletezés után.

A rendezési algoritmusok szerint csoportosított, helytelen megjelölések számának arányát az alábbi grafikon szemlélteti.



36. ábra: A megjelölt hamis algoritmus-állítás kombinációk aránya (helytelen válaszok) az egyes rendezési algoritmusok szerint csoportosítva

Annak érdekében, hogy megállapíthassuk, hogy a helytelen válaszok csökkenése szignifikáns-e, mindegyik rendezési algoritmusra külön-külön elvégeztük az előjelpróbát a diákok által igaznak megjelölt hamis állítások számán is.

Report

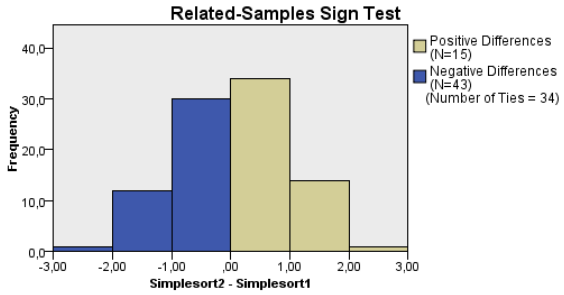
	Simplesort1	Simplesort2	Simplesort_diff
Median	1,00	0,00	0,000

	Bubblesort1	Bubblesort2	Bubblesort_diff
Median	1,00	0,00	0,000

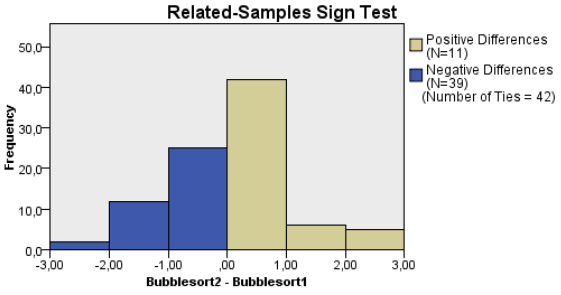
	Insertsort1	Insertsort2	Insertsort_diff
Median	1,00	1,00	0,000

	Minsort1	Minsort2	Minsort_diff
Median	0,00	0,00	0,000

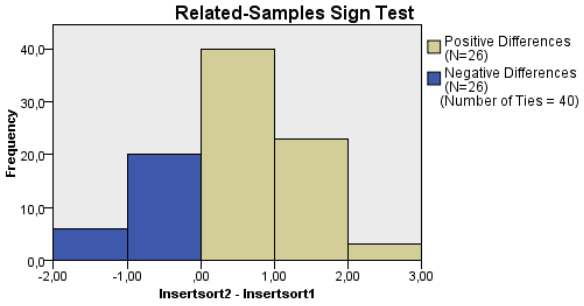
	Maxsort1	Maxsort2	Maxsort_diff
Median	0,00	0,00	0,000



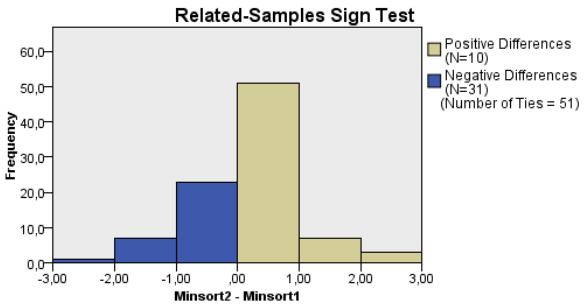
Total N	92
Test Statistic	15,000
Standard Error	3,808
Standardized Test Statistic	-3,545
Asymptotic Sig. (2-sided test)	,000



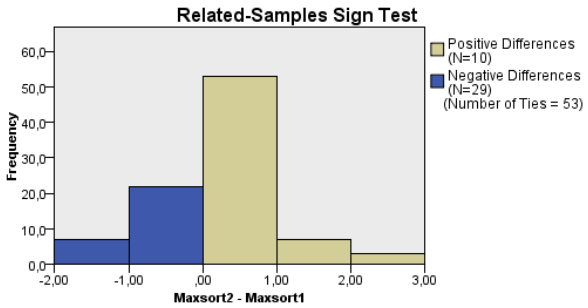
Total N	92
Test Statistic	11,000
Standard Error	3,536
Standardized Test Statistic	-3,818
Asymptotic Sig. (2-sided test)	,000



Total N	92
Test Statistic	26,000
Standard Error	3,606
Standardized Test Statistic	,000
Asymptotic Sig. (2-sided test)	1,000



Total N	92
Test Statistic	10,000
Standard Error	3,202
Standardized Test Statistic	-3,123
Asymptotic Sig. (2-sided test)	,002



Total N	92
Test Statistic	10,000
Standard Error	3,122
Standardized Test Statistic	-2,882
Asymptotic Sig. (2-sided test)	,004

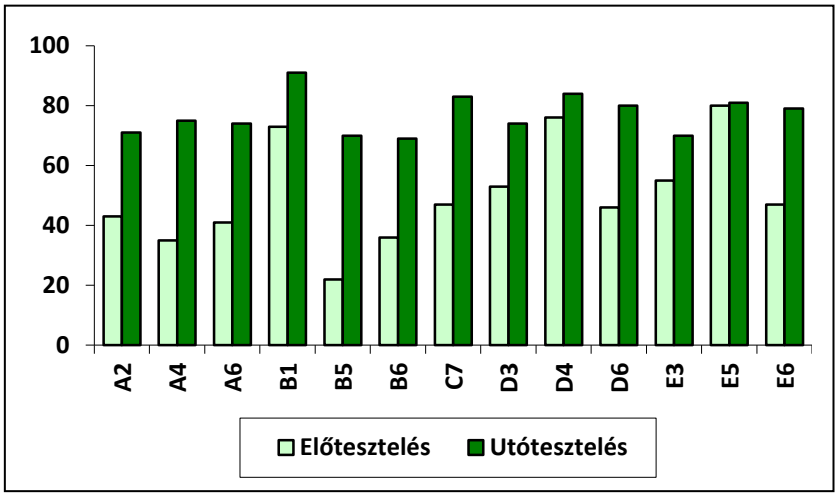
14. táblázat: Az előtesztelés (Simplesort1, Bubblesort1, Insertsort1, Minsort1, Maxsort1) és utótesztelés (Simplesort2, Bubblesort2, Insertsort2, Minsort2, Maxsort2) alatt az egyes diákok által az algoritmusokhoz helytelenül megjelölt állítások számán végzett előjelpróbák eredményei (SPSS Statistics, 2013)

A táblázatokból látható, hogy az előjelpróba az egyszerű cserés rendezés (simplesort), a buborékrendezés (bubblesort), a minimumkiválasztásos rendezés (minsort) és a maximumkiválasztásos rendezés (maxsort) algoritmusok esetében mutatott ki szignifikáns eredményt ($p < 0.05$). A diákok tehát szignifikánsan kevesebb helytelen választ jelöltek meg ezeknél az algoritmusoknál az animációkkal való kísérletezés után. A beszúró rendezés (insertsort) algoritmus esetében a változás nem szignifikáns ($p = 1.00$).

Az eddigiekben bemutatott összes eredményt figyelembe véve elmondhatjuk, hogy sikerült bebizonyítanunk a 2. hipotézist (az itt bemutatott animációk segítségével a diákok képesek felismerni az egyszerű rendezési algoritmusok lényeges lépéseit és az azok közti különbségeket) (Végh & Stoffová, 2017).

A továbbiakban mindegyik algoritmus-állítás kombinációra külön-külön is megvizsgáltuk a diákok által bejelölt X-ek számát. Főként azok az algoritmus-állítás kombinációk érdekeltek bennünket, ahol a megjelölések száma nem az elvárásainknak megfelelően növekedett vagy csökkent.

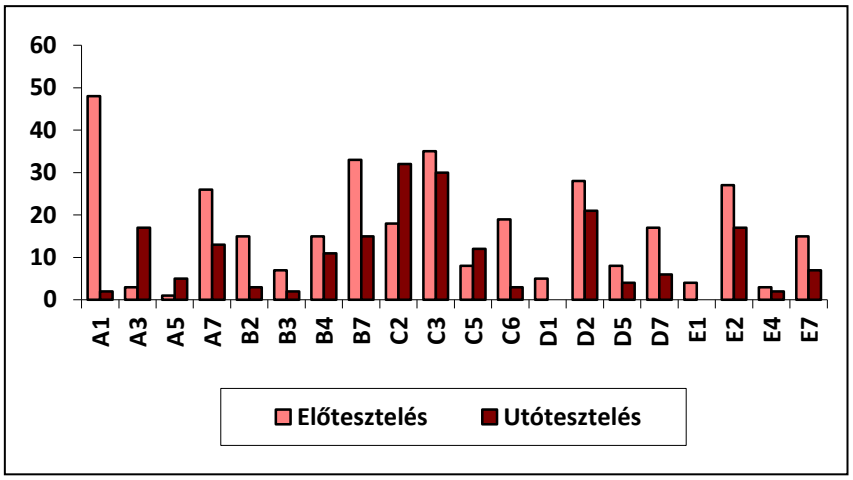
Az alábbi grafikon az igaz algoritmus-állítás kombinációkon összeszámolt válaszok számát szemlélteti, tehát a helyes válaszok számát.



37. ábra: Az igaz algoritmus-állítás kombinációkon összeszámolt válaszok száma

A grafikonból kiolvasható, hogy az animációkkal való kísérletezés után a diákok több igaz algoritmus-állítás kombinációt jelöltek be, a válaszok számában mindenhol növekedés tapasztalható.

Hasonló módon összeszámoltuk a hamis algoritmus-állítás kombinációkra kapott válaszok számát is. Ezt szemlélteti az alábbi grafikon.



38. ábra: A hamis algoritmus-állítás kombinációkon összeszámolt válaszok száma

Láthatjuk, hogy többségében csökkent a helytelen jelölések száma, azonban az A3-nál és a C2-nél nagyobb növekedés tapasztalható csökkenés helyett, ill. a C3-nál a csökkenés nem az elvárt mértékű. A következőkben ezek lehetséges okait próbáljuk meghatározni.

Az A3 állítás az egyszerű cserés rendezésre: „Mindig előbb végignézzük a rendezetlen elemeket, melyek közül kiválasztunk egyet, majd ezt az elemet cseréljük fel a tömbrész elején vagy végén levő elemmel.”

A C2 állítás a beszúró rendezésre: „Mindegyik elemet összehasonlítjuk az összes mögötte levővel.”.

A C3 állítás a beszúró rendezésre: „Mindig előbb végignézzük a rendezetlen elemeket, melyek közül kiválasztunk egyet, majd ezt az elemet cseréljük fel a tömbrész elején vagy végén levő elemmel.”

Ezek az állítások az adott algoritmusokra nem igazak. Azonban mindegyik állításnál, ha nem gondoljuk át részletesen az algoritmus működését és az egyes állításokat, igaznak tűnhetnek. Valószínűleg a diákok nem értelmezték részleteiben az itt említett állításokat és az animációknál sem gondoltak bele a részletes működésükbe. Az itt bemutatott animációknak azonban nem is az a céljuk, hogy részletesen bemutassák az egyes lépéseket. Ezen animációk után másik típusú, mikro-szintű animációk használata javasolt az oktatásban, melyeken a diákok megérthetik az egyes algoritmusok részletes működését (Hansen et al., 2002). Ilyen, mikro-szintű animációkkal fogunk foglalkozni a doktori értekezés 10. fejezetében.

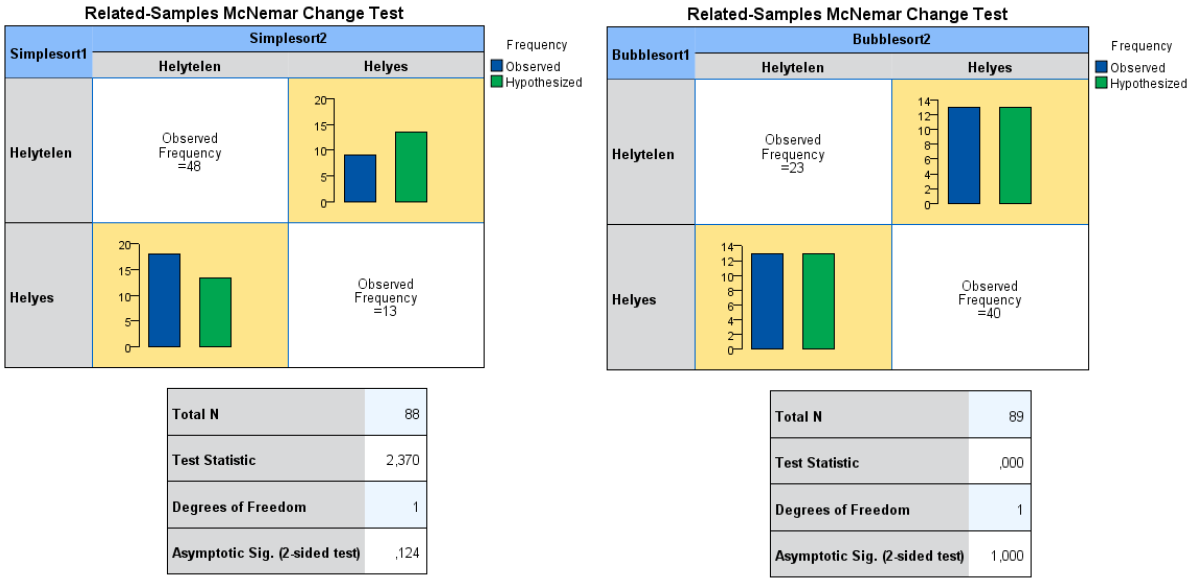
Az elő- és utótesztelés további részében kíváncsiak voltunk arra, hogy a diákok képesek-e a pszeudokódokhoz hozzápárosítani a rendezési algoritmusok neveit. Mivel az ilyen feladathoz

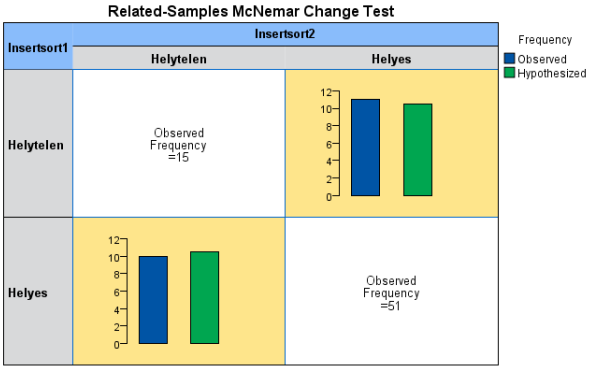
az algoritmusok részletes ismerete szükséges, nem feltételezzük, hogy a két teszt között szignifikáns javulás lesz.

	Egyszerű cserés rendezés (simplesort)	**Buborékrendezés (bubblesort)**	**Beszúró rendezés (insertion sort)**	**Minimumkivá-lasztásos rendezés (minsort)**	**Maximumkivá-lasztásos rendezés (maxsort)**
előtesztelés	64%	58%	66%	68%	67%
utótesztelés	72%	58%	67%	67%	66%

15. táblázat: Az algoritmusok neveihez helyesen párosított pszeudokódok aránya

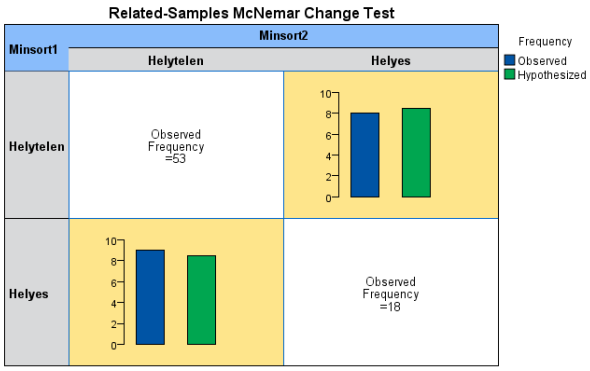
Mivel ennél a mérésnél dichotóm adataink vannak (0 = az algoritmushoz helytelenül hozzárendelt pszeudokód, 1 = az algoritmushoz helyesen hozzárendelt pszeudokód), ezért az előjelvizsgálat helyett McNemar-próbát végeztünk (SPSS Statistics, 2013).





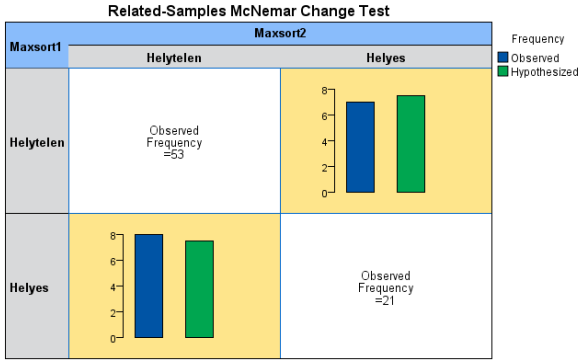
Total N	87
Test Statistic	,000
Degrees of Freedom	1
Asymptotic Sig. (2-sided test)	1,000
Exact Sig. (2-sided test)	1,000

1. The exact p-value is computed based on the binomial distribution because there are 25 or fewer records.



Total N	88
Test Statistic	,000
Degrees of Freedom	1
Asymptotic Sig. (2-sided test)	1,000
Exact Sig. (2-sided test)	1,000

1. The exact p-value is computed based on the binomial distribution because there are 25 or fewer records.



Total N	89
Test Statistic	,000
Degrees of Freedom	1
Asymptotic Sig. (2-sided test)	1,000
Exact Sig. (2-sided test)	1,000

1. The exact p-value is computed based on the binomial distribution because there are 25 or fewer records.

16. táblázat: Az előtesztelés (Simplesort1, Bubblesort1, Insertsort1, Minsort1, Maxsort1) és az utótesztelés (Simplesort2, Bubblesort2, Insertsort2, Minsort2, Maxsort2) alatt az algoritmusok pszeudokódokhoz való párosításának eredményein végzett McNemar-próbák (SPSS Statistics, 2013)

A táblázatokból látható, hogy az elő- és utótesztelés eredményei közötti változás nem szignifikáns egyik esetben sem (simplesort: $N = 88$, $\chi^2(1) = 2.370$, $p = 0.124$; bubblesort: $N = 89$, $\chi^2(1) = 0.000$, $p = 1.000$; insertion sort: $N = 87$, $p = 1.000$; minsort: $N = 88$, $p = 1.000$; maxsort: $N = 89$, $p = 1.000$).

Ezek alapján elmondhatjuk, hogy bár az animációk segítettek az egyes algoritmusok lényeges lépéseinek megértésében, az algoritmusok részleteiben való elsajátításában nem voltak hasznosak. Az ilyen fajta animációknak azonban nem is ez volt a céljuk, az algoritmusok részleteiben való megismeréséhez a 10. fejezetben bemutatott animációk használhatók.

Végezetül a 2014/15-ös akadémia évben végzett utótesztelés során kíváncsiak voltunk a hallgatók animációkkal kapcsolatos véleményére is. Ehhez a kérdőív kitöltésében részt vevő 39 diák egy-egy 10 fokozatú Likert-skálán jelölhette be az animációk érthetőségét, kezelhetőségét, és szemléletességét. Az összesített eredményeket az alábbi táblázat tartalmazza.

	Egyszerű cserés rendezés (simplesort)	**Buborékrendezés (bubblesort)**	**Beszúró rendezés (insertion sort)**	**Minimumkiválasztásos rendezés (minsort)**	**Maximumkiválasztásos rendezés (maxsort)**
	Érthetőség				
Mean:	9.72	9.72	9.59	9.64	9.64
Std. Dev.:	0.50	0.55	0.71	0.66	0.66
	Kezelhetőség				
Mean:	9.49	9.49	9.33	9.28	9.28
Std. Dev.:	0.75	0.75	1.05	1.08	1.08
	Szemléletesség				
Mean:	9.56	9.56	9.49	9.56	9.56
Std. Dev.:	0.74	0.74	0.84	0.74	0.74

17. táblázat: Az animációk értékelése a diákok által (10 fokozatú Likert-skálán)

A diákoknak tetszettek az interaktív animációk, magasan értékelték őket és a kérdőív megjegyzésébe többen is leírták, hogy szívesen tanulnának animációk segítségével más témaköröket is.

7.5 Tapasztalatok és módszertani tanácsok

Tapasztalataink azt mutatják, hogy az itt bemutatott interaktív algoritmusok segítségével a diákok képesek felismerni az egyes rendezési algoritmusok lényeges lépéseit és a köztük levő különbségeket. Az algoritmusok részletes megismerésére azonban ezen animációk nem elegendők.

Az itt felsorolt animációkat tehát a rendezési algoritmusokkal való ismerkedés legelső szakaszában ajánljuk felhasználni az oktatásban. A bemutatott kérdőívben található algoritmus-állításokat tartalmazó táblázathoz hasonló, nyomtatott táblázat kitöltése is javasolt az animációkkal való kísérletezés során, hiszen így a diákok jobban odafigyelnek az animációkban történő folyamatokra.

Ezek után, a rendezési algoritmusok oktatásának következő szakaszában, olyan interaktív animációk használata javasolt, ahol a diákok ugyanezekkel a rendezési algoritmusokkal

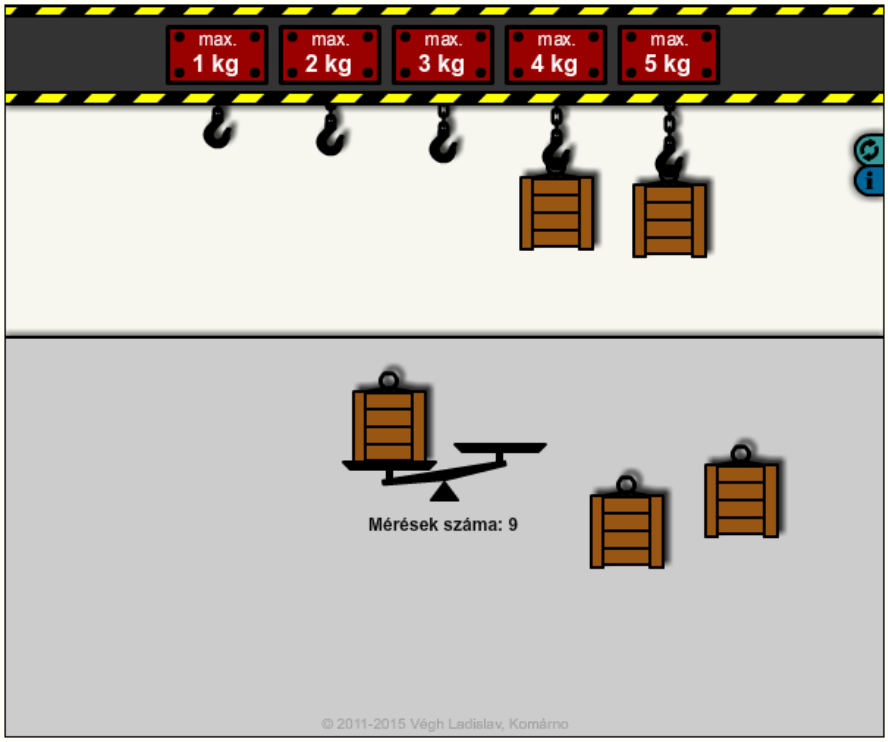
kísérletezhetnek, azonban már egy részletesebb, forráskóddal vagy pszeudokóddal is kiegészített animáció segítségével. Ilyen részletes, mikro-szintű animációkkal fogunk foglalkozni a munka 10. fejezetében.

8 Didaktikai játék: Ládák rendezése minimális számú összehasonlítással

Ebben a fejezetben egy másik általunk készített interaktív animációt – didaktikai játékot – mutatunk be. A játék célja: a lehető legkevesebb méréssel tömegük alapján növekvő sorrendbe rendezni a külsőre egyforma ládákat. Az interaktív animáció a http://anim.ide.sk/ladakrendezese.php weboldalon érhető el három nyelven (magyarul, szlovákul, angolul).

8.1 A szoftver felhasználói felületének bemutatása

A didaktikai játék három szintből áll. Az első szint a játék kezelésének megismerésére szolgál, ahol 3 láda növekvő sorrendbe való rendezése a cél. A második szint már bonyolultabb, itt 5 láda sorba rendezése a diákok feladata. Ezt követi végül a 7 ládából álló szint.



39. ábra: Didaktikai játék egy saját „effektív” rendezési algoritmus kifejlesztésére

A játék sikeres befejezéséhez a diákoknak össze kell hasonlítaniuk a ládák tömegeit egy kétkarú mérleg segítségével, majd megfelelő sorrendben felakasztani a ládákat a kampókra. Amennyiben valamelyik kampóra a diákok nehezebb ládát tesznek, mint amit az elbír, a kampó leszakad és a játékot elölről kell kezdeniük. Újrakezdés előtt azonban a diákoknak lehetőségük

van a sikertelen eredményhez vezetett folyamat egyes lépéseit végignézni, most már a ládákra írt tömegekkel együtt. Így rájöhetnek, hol vétettek hibát a gondolatmenetben.

40. ábra: Sikertelen próbálkozás után a ládák tömegei láthatóvá válnak és megjelennek a lépésenkénti visszajátszásra szolgáló nyomógombok

Azon célon túl, hogy a ládákat növekvő sorrendbe kell rendezni, egy másik célja is van a játéknak: a lehető legkevesebb számú mérés használata. A véletlen megoldást, tehát azt, hogy a diákok mérések nélkül, vagy korábbi mérésekből való kikövetkeztetések nélkül tippeljék meg az egyes ládák súlyát, teljes mértékben kizártuk a következő alfejezetben tárgyalt algoritmus segítségével. A diákoknak tehát a szükséges méréseket mindenképp el kell végezniük a játék sikeres befejezéséhez, azonban a jobb eredmény eléréséhez e mérések számát nem szabad túllépniük további redundáns (felesleges) mérésekkel.

8.2 A szoftver belső szerkezetének rövid jellemzése

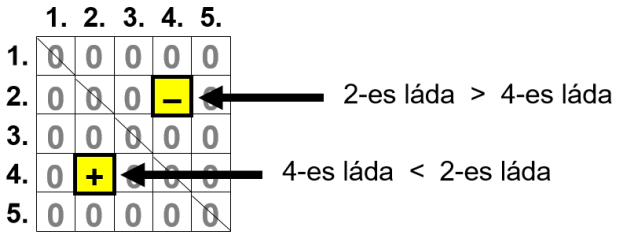
A didaktikai játékot Adobe Flash CS5 környezetben készítettük, ActionScript 3.0 programozási nyelv használatával. Ennek köszönhetően a játék könnyen beágyazható HTML oldalakba, például online tankönyvekbe, oktatással kapcsolatos blogokba.

A játékból sikerült teljes mértékben kizárnunk annak lehetőségét, hogy a diákok a szükséges mérések nélkül, próbálkozásokkal oldják meg a feladatot. Ehhez a ládák tömegeit

nem generáltuk ki a játék kezdetén, hanem folyamatosan, egymás után határoztuk meg őket egy-egy mérés alkalmával, vagy a játék sikertelen befejezésekor. Ennek köszönhetően bár a diákoknak úgy tűnik, hogy a ládák tömegei teljesen véletlenül vannak kigenerálva a játék elején, az összes szükséges mérést el kell végezniük a sikeres megoldáshoz (Végh, 2011b).

A ládák tömegeinek generálásához használt algoritmusban egy kétdimenziós tömböt (m) használtunk az elvégzett mérések megjegyzésére. Kezdetben a tömb összes eleme 0, amely azt jelenti, hogy a felhasználó egyelőre nem végzett el egyetlen mérést sem. Mindig, amikor a felhasználó összehasonlít két ládát, az eredményt ebbe a tömbbe jegyezzük be (amennyiben még nincs bejegyezve).

Példaként vegyük, hogy a játékos össze akarja hasonlítani a 2-es és 4-es sorszámú ládákat. Ilyenkor előbb megvizsgáljuk a mátrix m[2][4] elemét. Amennyiben ennek értéke 0, az azt jelenti, hogy az eddigi mérések alapján nem állapítható meg, hogy a két láda közül melyik a nehezebb. Ilyen esetben véletlenszám generátorral határozzuk meg a mérés eredményét, majd az m[2][4] tömbelemet beállítjuk „-” értékre (ha a 2-es sorszámú láda nehezebb, mint a 4-es) vagy „+” értékre (ha a 2-es sorszámú láda könnyebb, mint a 4-es). Ezután a tömb m[4][2] elemét is beállítjuk az ellentétes értékre.



41. ábra: A 2-es és 4-es láda közötti reláció beírása az m mátrixba (m[2][4]=”-” és m[4][2]=”+”)

Sok esetben, két láda közötti összehasonlításból kikövetkeztethetők további ládák közötti relációk is. Például, ha az alábbi mérések már be vannak jegyezve a mátrixba:

- 1-es láda < 2-es láda,
- 5-ös láda < 3-as láda,

és ez után az alábbi relációt szeretnénk bejegyezni:

- 2-es láda < 5-ös láda,

akkor az alábbi relációkat is be kell jegyeznünk a mátrixba, hiszen ezek logikailag következnek az eddigi mérésekből:

- 1-es láda < 5-ös láda,
- 1-es láda < 3-as láda,
- 2-es láda < 3-as láda.

Az ilyen, logikailag kikövetkeztethető mérések mátrixba írását az alábbi **compareAndSet** függvénnyel és **setLighter** rekurzív függvénnyel végeztük el.

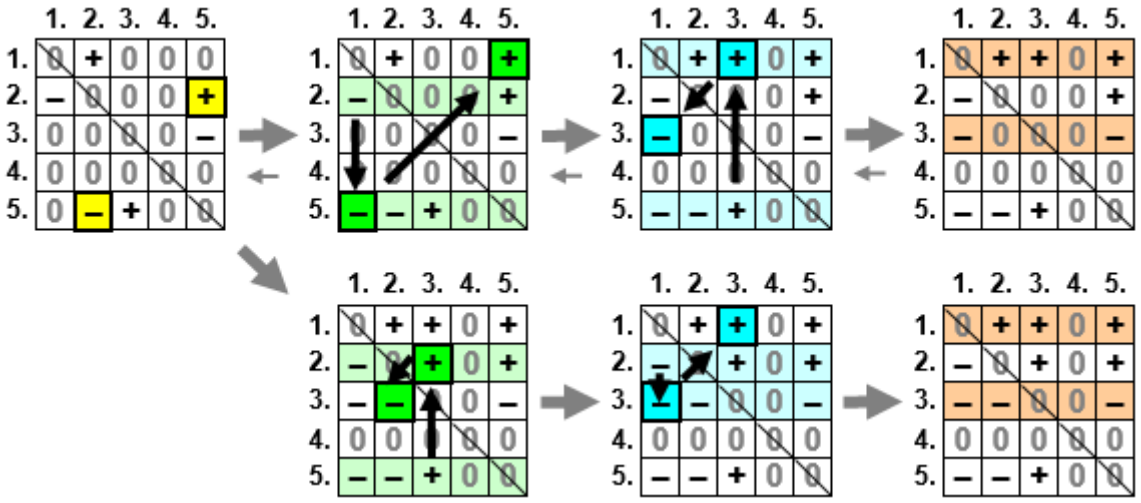
```
// Amennyiben még nem határoztuk meg az n1 és n2 ládák közötti relációt, ez a függvény
// kigenerálja ezt a relációt, majd beírja a mátrixba a setLighter függvény meghívásával.
private function compareAndSet(n1:uint, n2:uint):void
{
  if (m[n1][n2]=="0")
  {
       var i:uint;
       if (Math.random()<0.5)
       {
            // n2 < n1
            setLighter (n2,n1);
       } else {
            // n1 < n2
            setLighter(n1,n2);
       }
  }
}
```

```
// Ez a függvény beírja a mátrixba az n1<n2 relációt, majd rekurzívan meghívja saját
// magát azokra a ládákra, melyek között a relációk logikailag kikövetkeztethetők.
private function setLighter(n1:uint, n2:uint):void
{
  m[n1][n2]="+"; m[n2][n1]="-";
  for (var i:uint=1; i<=num; i++)
  {
       if (m[n1][i]=="-") { setLighter(i,n2); }
       if (m[n2][i]=="+") { setLighter(n1,i); }
  }
}
```

A **setLighter(K,L)** rekurzív függvény, amely a K<L relációt írja be a mátrixba, az alábbi módon működik (42. ábra):

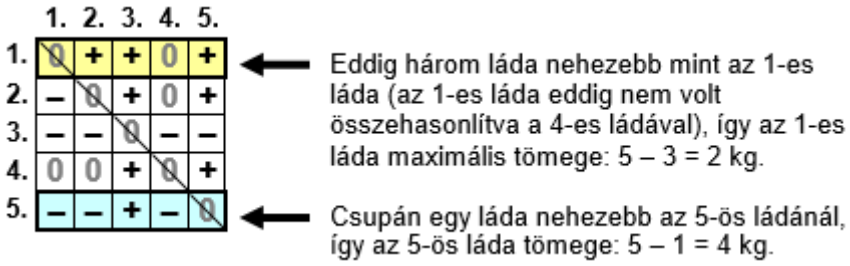
- Előbb beállítjuk a következő tömbelemek értékét: **m[K,L]=”+”** és **m[L,K]=”–”**.
- Ezután megvizsgáljuk a **K.** sort. Ha létezik **J**, melyre **J. láda < K. láda** igaz, akkor az is igaz, hogy **J. láda < L. láda**. Minden ilyen **J**-re meghívjuk a **setLighter(J,L)** rekurzív függvényt.

- Végül megvizsgáljuk az **L.** sort. Ha létezik **J**, melyre az **L. láda < J. láda** igaz, akkor az is igaz, hogy **K. láda < J. láda**. Minden ilyen **J**-re meghívjuk a **setLighter(K,J)** rekurzív függvényt.



42. ábra: A ládák tömege közötti, logikailag kikövetkeztethető relációk beállítása a mátrixban a „setLighter” rekurzív függvény használatával (a 2-es és 5-ös ládák közötti reláció beírása után)

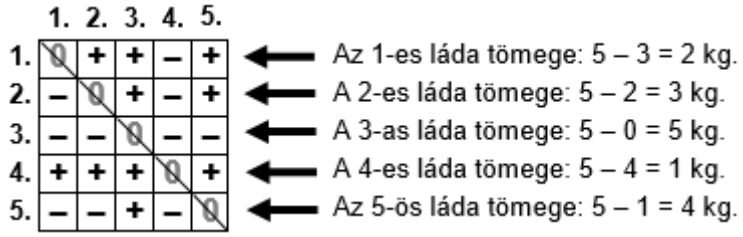
Az itt ismertetett algoritmussal elkészült mátrix segítségével könnyen meghatározható az egyes ládák maximális tömege. Ehhez mindössze össze kell számolnunk azon ládákat, melyek nehezebbek, mint az adott láda. Tehát csupán meg kell számlálnunk a „+” jeleket az adott ládához tartozó sorban.



43. ábra: A ládák maximális tömegeinek meghatározása a mátrixból

Minden kampónak a játékban van egy maximális teherbírása. Ha a játékos egy olyan ládát tesz az adott kampóra, amely tömege a fenti módon meghatározva több lehet, mint a kampó teherbírása, akkor a kampó leszakad és a játék sikertelenül ér véget. Ilyen esetben a kampóra akasztott láda sorát feltöltjük a mátrixban „–” jelekkel a **setLighter** rekurzív függvény segítségével (tehát a láda tömegének a lehetséges legnagyobb értéket vesszük). Végezetül kigeneráljuk a mátrixban azon ládák közötti relációkat, amelyek még nincsenek meghatározva.

Egy ilyen módon kigenerált mátrixból már könnyedén meghatározhatók a ládák pontos tömegei. Ezt szemlélteti az alábbi ábra.



44. ábra: A játék végeztével kigenerált teljes mátrix és az ebből meghatározott ládatömegek

8.3 Kísérlet bemutatása

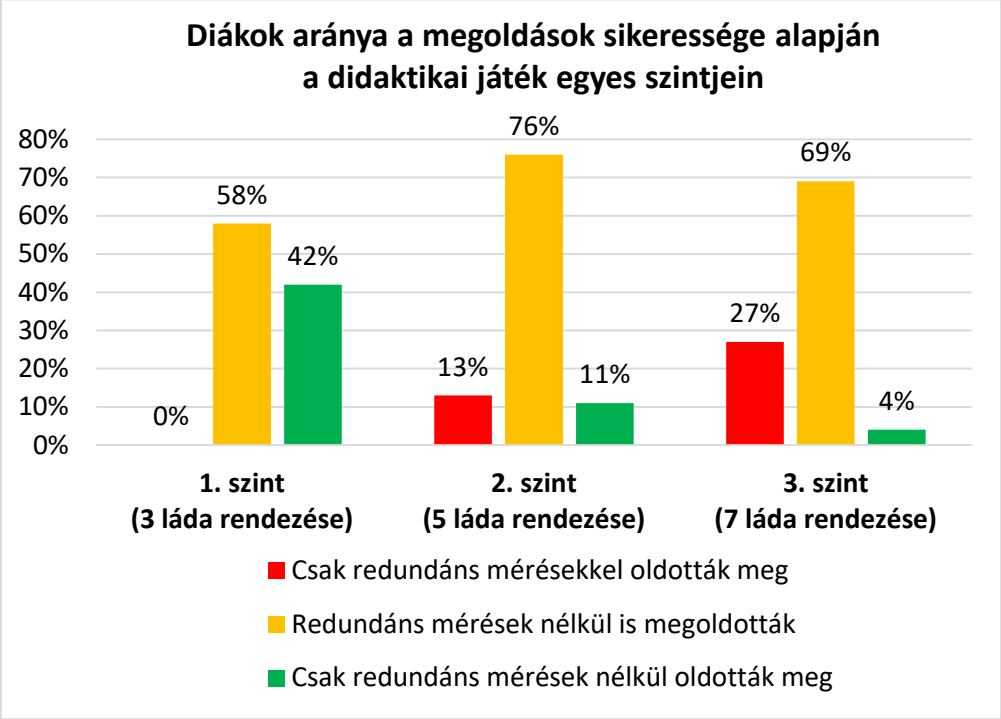
A didaktikai játék segítségével a 2014/15-ös és 2015/16-os akadémiai év nyári szemeszterében pedagógiai kísérletet végeztünk a Selye János Egyetemen, melyben 99 elsős informatika szakos hallgató vett részt: 50 a 2014/15-ös évben és 49 a 2015/16-os évben (Végh, 2016e; Végh & Stoffová, 2016). A hallgatók a kísérlet előtt már találkoztak egyszerű rendezési algoritmusokkal, melyek bonyolultsága $O(n^2)$, azonban $O(n.\log_2 n)$ bonyolultságú algoritmusokat még nem vettek az „Algoritmusok és programozás” óra keretén belül. Annak ellenére, hogy a diákok nem foglalkoztak még az algoritmusok hatékonyságának vizsgálatával és a redundáns (felesleges) összehasonlítások kiszűrésével, a kísérlet során megkértük őket, hogy próbáljanak kitalálni egy redundáns összehasonlítások nélküli folyamatot a ládák rendezésére. A feladatuk tehát az volt, hogy lehető legkevesebb mérést végezzenek el a rendezés során. A felesleges mérések számát (melyek előző mérésekből logikailag kikövetkeztethetők) a program kiírta a rendezés végén.

A kísérlet alatt végzett összehasonlítások számának elmentéséhez létrehoztunk egy MySQL adatbázist, ennek szerkezetét a doktori értekezés 11. melléklete tartalmazza. Minden felhasználó az http://anim.ide.sk/ladakrendezese.php weboldal megnyitásakor egy egyedi azonosítót kapott, amelyet az adatbázisban is tároltunk. A diákokat megkértük, hogy ezt az azonosítót írják fel a kiosztott kérdőívekre, amely a doktori értekezés 12. mellékletében található. Ilyen módon a kérdőív teljesen mértékben anonim volt, a diákok által adott válaszokat mégis össze tudtuk kapcsolni az adatbázisba elmentett adatokkal, ill. kiszűrni az adatbázisból a tesztelt diákokhoz tartozó adatokat.

8.4 Eredmények kiértékelése és elemzése

Az adatbázisból kinyert adatok és a diákok kérőívre adott válaszaik a doktori értekezés 13. mellékletében tekinthetők meg.

Az összes diáknak sikerült valamilyen megoldást találnia a probléma megoldására. Voltak, akiknek csak redundáns mérések segítségével sikerült megoldani a feladatot, azonban a többség megoldotta redundáns mérések nélkül is. Olyanok is voltak, akiknek elsőre sikerült megoldani a feladatot redundáns mérések nélkül. Ezek arányát az alábbi grafikon szemlélteti.



45. ábra: Diákok aránya a feladat megoldásának sikeressége alapján az egyes szinteken

Továbbá megvizsgáltuk azt is, hogy a diáknak a sikeres próbálkozások alatt sikerült-e lecsökkenteni a redundáns mérések számát, esetleg kitalálni egy olyan algoritmust a ládák rendezésére, amely egyáltalán nem végez redundáns méréseket. Ilyen ismert rendezési algoritmusok többek között az $O(n.\log_2 n)$ bonyolultságú algoritmusok, tehát pl. a quicksort vagy a mergesort. Azonban az $O(n^2)$ bonyolultságú beszúró rendezés algoritmusa sem végez redundáns összehasonlításokat, ill. a buborékrendezésnél is lecsökkenthető a felesleges összehasonlítások száma.

Mindegyik szinten (3-ládás, 5-ládás, 7-ládás) megvizsgáltuk, hogy az egyes sikeres próbálkozások után a diákok által végzett redundáns mérések száma csökkent-e.

Az alábbi táblázatokban Level1 jelenti az első szintet (3 láda), Level2 a második szintet (5 láda), és Level3 a harmadik szintet (7 láda). Mindegyik szinten a felhasználók első 6 sikeres próbálkozását vettük figyelembe (Attempt1, .., Attempt6). Feltételeztük, hogy a diákok minél többször próbálkoznak, annál kevesebb redundáns mérést használnak a feladat megoldására.

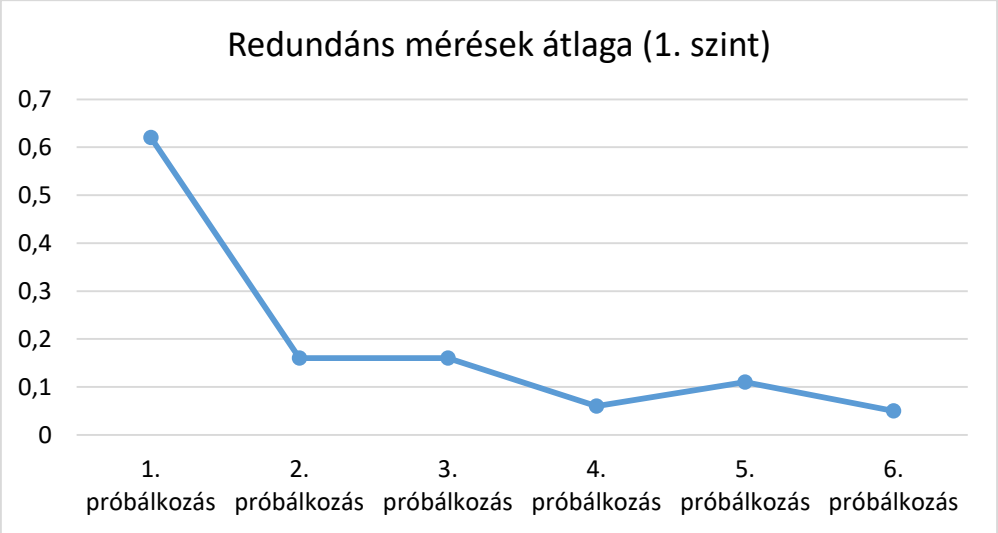
Az alábbi táblázat az első szint (3 láda) sikeres megoldása során végzett redundáns mérések számára vonatkozó statisztikai eredményeket tartalmazza.

Descriptive Statistics

	Mean	Std. Deviation	Median	N
Level1Attempt1	0,62	1,251	0	99
Level1Attempt2	0,16	0,397	0	99
Level1Attempt3	0,16	0,467	0	99
Level1Attempt4	0,06	0,246	0	94
Level1Attempt5	0,11	0,313	0	83
Level1Attempt6	0,05	0,229	0	73

18. táblázat: Az első szint (3 láda) megoldásakor a sikeres próbálkozások során végzett redundáns mérések számával kapcsolatos statisztikai adatok

A sikeres megoldások során elvégzett redundáns mérések átlagait grafikonon is szemléltettük (46. ábra), melyből látszik, hogy összességében csökkent a redundáns mérések száma a feladat megoldása során.



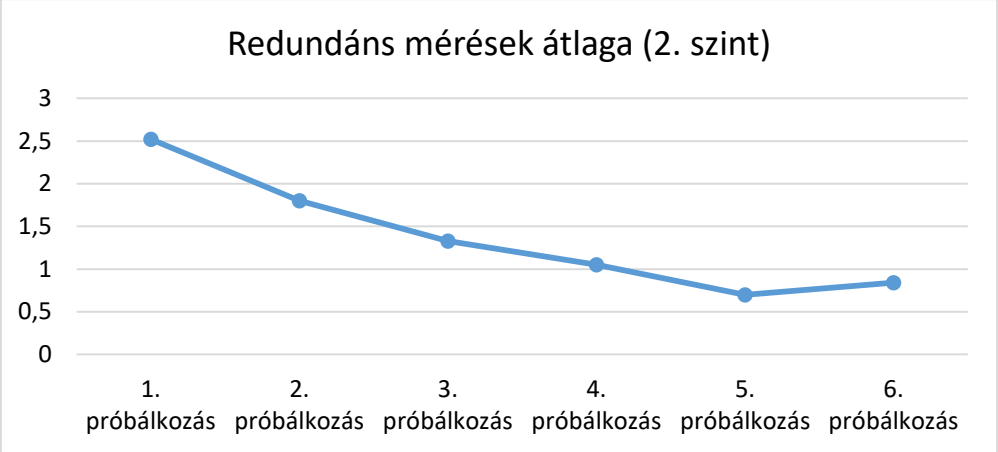
46. ábra: A redundáns mérések átlagai az első hat sikeres próbálkozás során az első szint (3 láda) megoldásakor

Az alábbi táblázat a második szint (5 láda) sikeres megoldása során végzett redundáns mérések eredményeit tartalmazza, amelyeket grafikonon is szemléltettünk.

Descriptive Statistics

	Mean	Std. Deviation	Median	N
Level2Attempt1	2,52	2,894	1	99
Level2Attempt2	1,80	2,134	1	98
Level2Attempt3	1,33	1,706	0	97
Level2Attempt4	1,05	1,615	0	91
Level2Attempt5	0,70	1,174	0	80
Level2Attempt6	0,84	1,542	0	68

19. táblázat: A második szint (5 láda) megoldásakor a sikeres próbálkozások során végzett redundáns mérések számával kapcsolatos statisztikai adatok



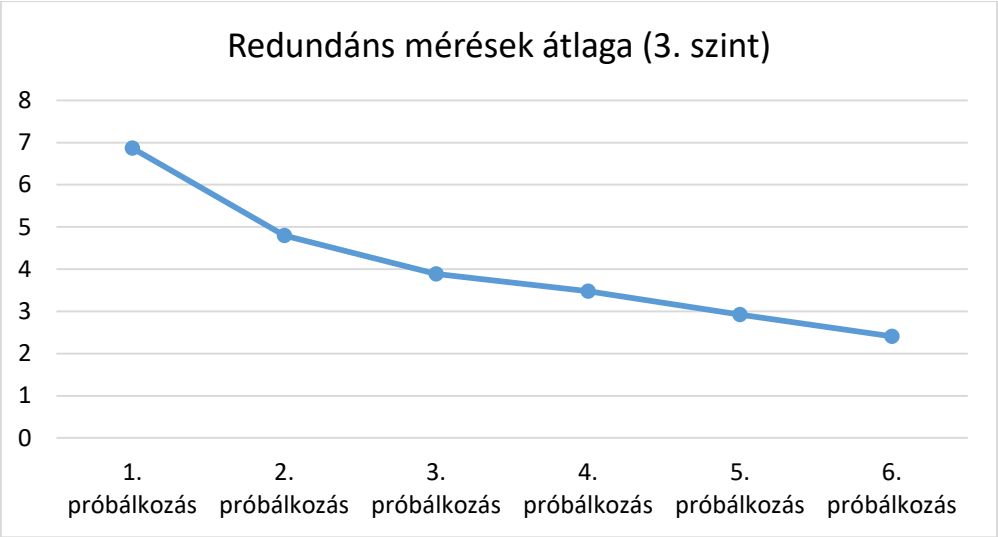
47. ábra: A redundáns mérések átlagai az első hat sikeres próbálkozás során a második szint (5 láda) megoldásakor

Az következő táblázat a harmadik szint (7 láda) sikeres megoldása során végzett redundáns mérések eredményeit tartalmazza, amelyeket szintén szemléltettünk grafikonokon.

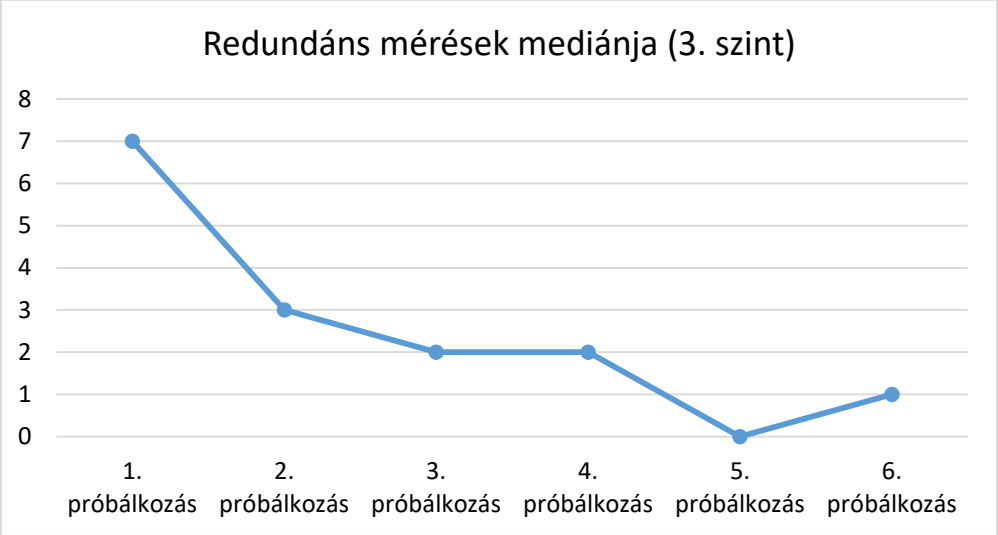
Descriptive Statistics

	Mean	Std. Deviation	Median	N
Level3Attempt1	6,87	5,452	7	99
Level3Attempt2	4,80	4,815	3	97
Level3Attempt3	3,89	4,503	2	92
Level3Attempt4	3,48	4,435	2	83
Level3Attempt5	2,92	4,574	0	72
Level3Attempt6	2,41	3,836	1	61

20. táblázat: A harmadik szint (7 láda) megoldásakor a sikeres próbálkozások során végzett redundáns mérések számával kapcsolatos statisztikai adatok



48. ábra: A redundáns mérések átlagai az első hat sikeres próbálkozás során a harmadik szint (7 láda) megoldásakor



49. ábra: A redundáns mérések mediánjai az első hat sikeres próbálkozás során a harmadik szint (7 láda) megoldásakor

A harmadik szint (7 láda) megoldásánál már jelentősebb változás figyelhető meg. Mivel az egymás utáni sikeres próbálkozások mediánjaiban is látható csökkenés, annak megállapítására, hogy ez a csökkenés szignifikáns-e, Friedman tesztet (Related-Samples Friedman's Two-Way Analysis of Variance by Ranks, SPSS Statistics, 2013) végeztünk. Azért erre a tesztre esett a választásunk az egyszempontos összetartozó mintás varianciaanalízis (One-Way Repeated Measures ANOVA) helyett, mert az adatok nem normál eloszlásúak (21. táblázat), ami az egyszempontos összetartozó mintás varianciaanalízis elvégzéséhez szükséges feltétel.

Test of Normality

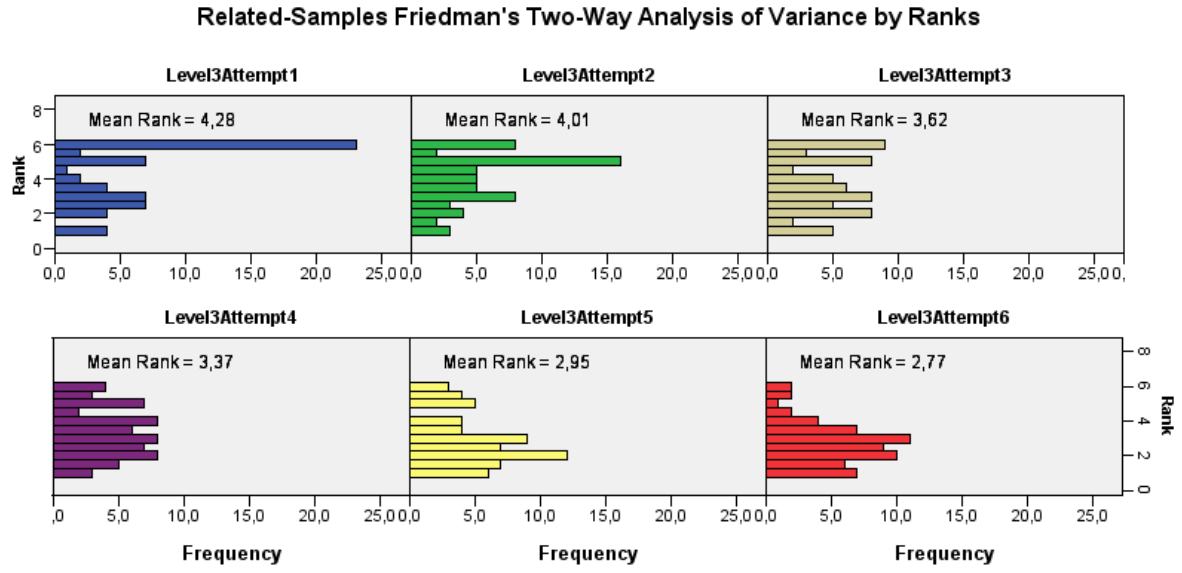
	Shapiro-Wilk		
	Statistic	df	Sig.
Level3Attempt1	0,875	61	0,000
Level3Attempt2	0,870	61	0,000
Level3Attempt3	0,842	61	0,000
Level3Attempt4	0,770	61	0,000
Level3Attempt5	0,700	61	0,000
Level3Attempt6	0,665	61	0,000

21. táblázat: A Shapio-Wilk próbák eredménye: az adatok nem normál eloszlásúak (SPSS Statistics, 2013)

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The distributions of Level3Attempt1, Level3Attempt2, Level3Attempt3, Level3Attempt4, Level3Attempt5 and Level3Attempt6 are the same.	Related-Samples Friedman's Two-Way Analysis of Variance by Ranks	,000	Reject the null hypothesis.

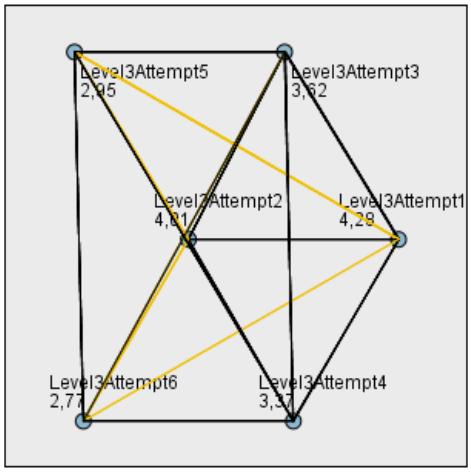
Asymptotic significances are displayed. The significance level is ,05.



Total N	61
Test Statistic	35,560
Degrees of Freedom	5
Asymptotic Sig. (2-sided test)	,000

50. ábra: A Friedman teszt eredménye (SPSS Statistics, 2013)

Pairwise Comparisons



Each node shows the sample average rank.

Sample1-Sample2	Test Statistic	Std. Error	Std. Test Statistic	Sig.	Adj.Sig.
Level3Attempt6-Level3Attempt5	,180	,339	,532	,594	1,000
Level3Attempt6-Level3Attempt4	,598	,339	1,766	,077	1,000
Level3Attempt6-Level3Attempt3	,852	,339	2,516	,012	,178
Level3Attempt6-Level3Attempt2	1,238	,339	3,654	,000	,004
Level3Attempt6-Level3Attempt1	1,508	,339	4,452	,000	,000
Level3Attempt5-Level3Attempt4	,418	,339	1,234	,217	1,000
Level3Attempt5-Level3Attempt3	,672	,339	1,984	,047	,709
Level3Attempt5-Level3Attempt2	1,057	,339	3,121	,002	,027
Level3Attempt5-Level3Attempt1	1,328	,339	3,920	,000	,001
Level3Attempt4-Level3Attempt3	,254	,339	,750	,453	1,000
Level3Attempt4-Level3Attempt2	,639	,339	1,887	,059	,887
Level3Attempt4-Level3Attempt1	,910	,339	2,686	,007	,109
Level3Attempt3-Level3Attempt2	,385	,339	1,137	,255	1,000
Level3Attempt3-Level3Attempt1	,656	,339	1,936	,053	,794
Level3Attempt2-Level3Attempt1	,270	,339	,798	,425	1,000

Each row tests the null hypothesis that the Sample 1 and Sample 2 distributions are the same.
Asymptotic significances (2-sided tests) are displayed. The significance level is ,05.

51. ábra: Páronkénti post hoc összehasonlítások Bonferroni-korrekcióval (SPSS Statistics, 2013)

A Friedman teszt eredményéből (50. ábra) kiolvasható, hogy az egymás utáni sikeres próbálkozások során végzett redundáns mérések számainak eloszlása egymástól szignifikánsan különbözik, $\chi^2(5) = 35.560,\ p < 0.0005$. Annak felderítésére, hogy a felesleges mérések számában megfigyelhető szignifikáns változások melyik sikeres próbálkozások között mutathatók ki, páronkénti összehasonlításokat végeztünk a többszörös összehasonlításhoz használt Bonferroni-korrekcióval (Friedman test with pairwise comparisons, SPSS Statistics, 2013). Ennek eredménye látható a 51. ábrán.

A post hoc analízis szignifikáns különbségeket mutatott ki az **első (Mdn = 7) és ötödik (Mdn = 0) (p = 0.001)**, az **első (Mdn = 7) és hatodik (Mdn = 1) (p < 0.0005)**, a **második (Mdn = 3) és ötödik (Mdn = 0) (p = 0.027)**, és a **második (Mdn = 3) és hatodik (Mdn = 1) (p = 0.004)** sikeres próbálkozások közötti felesleges mérések számában.

Ezzel sikerült bebizonyítanunk a 3. hipotézist (az itt bemutatott interaktív animációval a hallgatók képesek olyan rendezési algoritmust megalkotni, amelyben az eredetileg általuk megalkotott rendezési algoritmushoz képest lecsökken az összehasonlítások száma).

A pedagógiai kísérlet során az is érdekelt bennünket, hogy a diákok milyen algoritmust alkalmaztak a mérések számának csökkentésére, ill. milyen algoritmussal sikerült nekik redundáns mérések nélkül megoldani a feladatot. Ennek érdekében egyrészt a kísérlet alatt megfigyeltük néhány diák megoldásmenetét, másrészt megkértük a diákokat, hogy a kérdőív utolsó kérdésében írják le röviden az általuk alkalmazott algoritmus menetét saját szavaikkal.

A ládarendezés néhány lehetséges megoldását az alábbi videók szemléltetik:

1. **Bubblesort:** https://youtu.be/47Ikl5wyRMM

   A rendezés során sorban felrakjuk a ládákat a kétkarú mérlegre. A mérlegről minden összehasonlítás után a könnyebb ládát vesszük le, így a végén a legnehezebb láda marad a mérlegen. A folyamatot ezek után megismételjük a megmaradt ládákra.

2. **Bubblesort + jelölés, 1. verzió:** https://youtu.be/YAf1Rgj0z2A

   Hasonlóan az előző módszerhez, itt is sorban felraktuk a kétkarú mérlegre a ládákat, majd mindig a könnyebbet vesszük le. A levételnél azonban figyelünk arra, hogy a mérleg melyik oldaláról vettük le a ládát. Amennyiben a ládát ugyanarról az oldalról vettük le kétszer (vagy többször) egymás után, a másodjára (harmadjára, stb.) levett ládát még össze kell hasonlítanunk az előtte levett ládákkal, ezért ezt az előző fölé, külön csoportba tesszük, ezzel megjelölve a ládát. Miután megtaláltuk a

legnehezebb ládát, figyelembe vesszük a külön csoportosított (megjelölt) ládákat és elvégezzük velük a még hiányzó összehasonlításokat. Bár az algoritmus bonyolultabb a jelölések miatt, az összehasonlítások száma minimális (nincsenek redundáns mérések).

Bubblesort + jelölés, 2. verzió: https://youtu.be/zbodybqAFJ8

Az elv ugyanaz, mint az az 1. verziónál, a különbség csupán annyi, hogy nem mindig a legkönnyebbet, hanem a legnehezebbet vesszük le a mérlegről, és ezt szükség szerint megjelöljük (külön csoportba rakjuk). Így a szükséges összehasonlítások után először nem a legnehezebb, hanem a legkönnyebb ládát kapjuk meg.

3. **Quicksort:** https://youtu.be/eqO1x4Fj5RM

   A ládák összehasonlítása a gyorsrendező algoritmus alapján. Kiválasztunk egy tetszőleges ládát és ezzel hasonlítjuk össze a többit. A kiválasztott ládánál könnyebbeket az egyik csoportba (balra), nehezebbeket a másik csoportba (jobbra) rakjuk. Ezek után a folyamatot elvégezzük mindkét csoportra külön-külön.

4. **Mergesort:** https://youtu.be/zLpZWiYT7Fo

   A ládák összehasonlítása az összefésülő algoritmus alapján. Előbb maximum 2-ládás rendezett csoportokat alakítunk ki, majd ezeket összefésülve kialakulnak maximum 4-ládás csoportok, melyeket szintén összefésüléssel rendezünk maximum 8-ládás csoportokba, stb. (a didaktikai játékban maximum 7-ládás rendezett csoportba rendeztük őket).

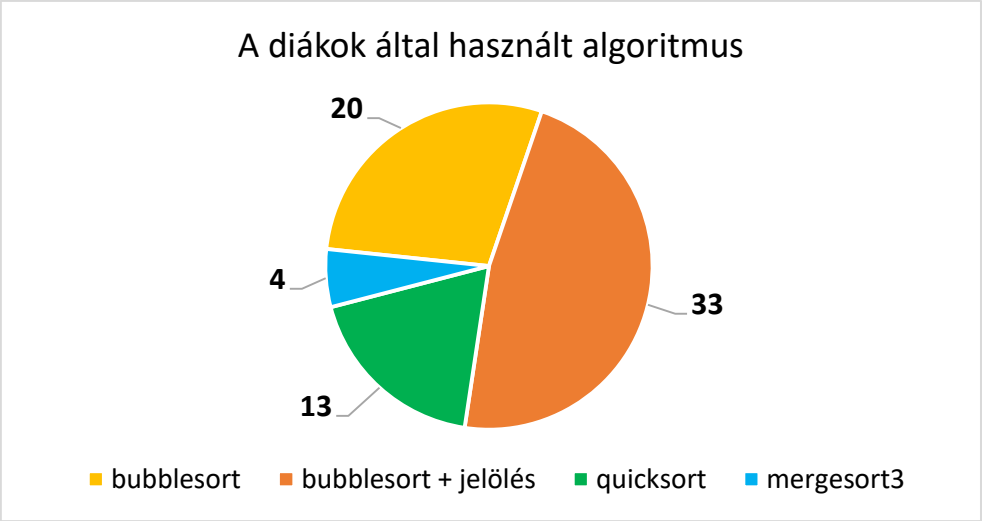
5. **3-4-elemű csoportok összefésülése:** https://youtu.be/FWdqbmSs1kw

   A ládák rendezése 3-as csoportokba minimális számú összehasonlítással (2 vagy 3 mérés), majd a kimaradó láda beszúrása valamelyik csoportba. Végül, a két csoport összefésülése egy rendezett sorrá.

A megfigyeléseink alapján a diákok többsége kezdetben az első algoritmussal próbálkozott (bubblesort), azonban hamar rájöttek, hogy így több olyan mérést is el kell végezniük, melyek már logikailag kikövetkeztethetők korábbi mérésekből. Ezek után próbálták valamilyen módon minimalizálni a mérések számát. Így jutottak el többen is a második algoritmus valamelyik verziójához (bubblesort + jelölés).

Voltak diákok, akik végül a harmadik algoritmus (quicksort) segítségével oldották meg a feladatot. A negyedik algoritmussal (mergesort) egyik diák sem próbálkozott, ennek kissé módosított változatával (3-4 elemű csoportok összefésülése - mergesort3) azonban néhány diáknak sikerült megoldani a feladatot redundáns mérések nélkül.

A diákok a feladatlap 12. kérdésére leírták az általuk használt módszert. Ezeket áttanulmányozva mindegyikhez hozzárendeltük az említett algoritmusok valamelyikét. A felmérésből kiderült, hogy a diákok szignifikáns többsége (47%) a második algoritmust (bubblesort + jelölés) használta: $\chi^2(3, N = 70) = 25.657, p < 0.0005$.



52. ábra: A diákok által használt algoritmus (a kérdőív 12. kérdésére adott válaszok alapján)

Fontos megjegyeznünk, hogy a diákok által továbbfejlesztett buborékrendezés jóval kevesebb összehasonlítást végez, mint az eredeti buborékrendezés vagy a közismert továbbfejlesztett változat, továbbá nincs benne redundáns összehasonlítás (melyek logikailag kikövetkeztethetők a korábbi összehasonlításokból). Az összehasonlítások száma azonban még így is jóval több, mint a gyorsrendező algoritmusnál (a buborékrendezés elvéből adódóan). Továbbá, a diákok által továbbfejlesztett buborékrendező algoritmusban a cserék száma nem csökkent, tehát a gyakorlatban egy tömb rendezésekor az algoritmus ugyanannyi cserét végez, mint az eredeti buborékrendezés. Mindezeket az alábbi táblázat is szemlélteti, amely 10000 véletlen szám rendezésénél végrehajtott összehasonlítások és cserék számát tartalmazza.

	összehasonlítások száma	cserék száma
buborékrendezés (bubblesort1 függvény)	49995000	24837078
továbbfejlesztett buborékrendezés (bubblesort2 függvény)	49940418	24837078
diákok által kigondolt buborékrendezés (bubblesort3 függvény)	24847071	24837078
gyorsrendezés (quicksort függvény)	105265	33469

22. táblázat: 10000 véletlen szám rendezéséhez felhasznált összehasonítások és cserék száma a buborékrendezés különböző változatainál és a gyorsrendező algoritmusnál (14. mellékletben található Java program futási eredménye)

A diákok által kitöltött kérdőív tartalmazott 6 logikai feladatot is, mellyel azt szerettük volna felmérni, hogy a didaktikai játékkal való kísérletezés után képesek-e megoldani ezeket a feladatokat. Mindegyik feladat szorosan összefüggött a redundáns mérések meghatározásával. A diákok többségének sikerült megoldani a feladatokat, az általuk bejelölt helyes válaszok aránya az alábbi táblázatban található.

1. feladat	2. feladat	3. feladat	4. feladat	5. feladat	6. feladat
96%	95%	88%	96%	84%	75%

23. táblázat: A kérdőíven található logikai feladatok (5.-10. kérdés) megoldásainak eredményessége

Érdekes jövőbeli kutatás lehetne annak megállapítása, hogy mennyire segített ilyen és ezekhez hasonló típusú logikai feladatok megoldásában a didaktikai játék.

Végezetül kíváncsiak voltunk a hallgatók játékkal kapcsolatos véleményére. Ehhez a diákok egy-egy 10 fokozatú Likert-skálán jelölhették be a didaktikai játék érthetőségét, kezelhetőségét, és szemléletességét. Az összesített eredményeket az alábbi táblázat tartalmazza.

	Összességében	Érthetőség	Kezelhetőség	Grafika
Mean:	8.53	9.49	9.44	9.14
Std. Dev.:	1.60	1.10	1.19	1.31

24. táblázat: A didaktikai játék értékelése a diákok által (10 fokozatú Likert-skálán)

A diákoknak tetszett a játék, magasan értékelték. Az óra végén többen is jelezték, hogy szívesen látnának hasonló didaktikai játékokat az oktatásban.

8.5 Tapasztalatok és módszertani tanácsok

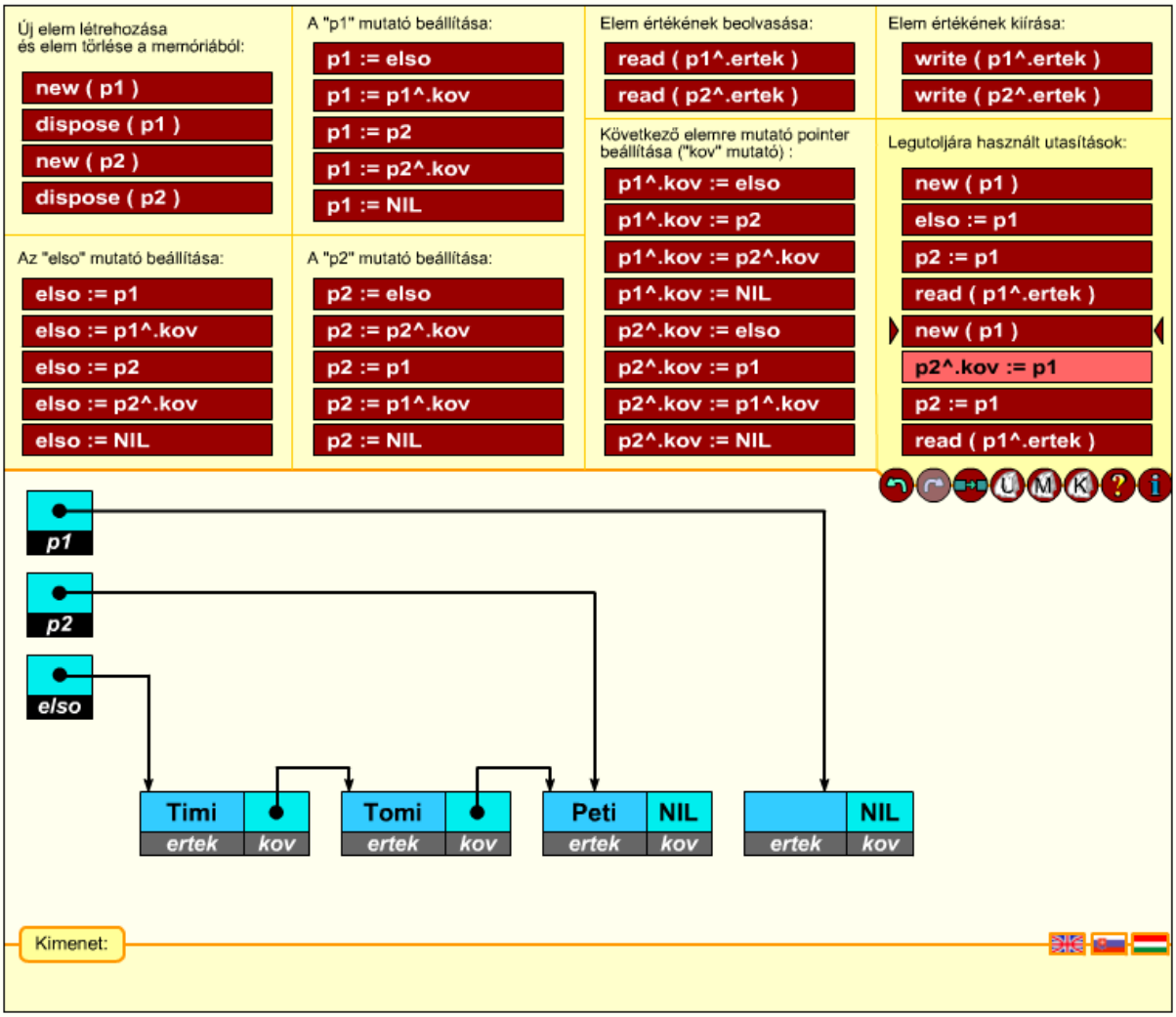
Tapasztalataink szerint a diákok kedvelték a játékot, többségüknek sikerült megoldani a feladatot redundáns mérések nélkül. Az is kiderült, hogy a diákok fokozatosan fejlesztették az általuk megalkotott algoritmusokat. Bár a mérések (összehasonlítások) számát sikerült így a minimálisra csökkenteniük, a legtöbb diák algoritmusa mégsem lenne elég effektív a gyakorlatban (pl. gyorsrendező algoritmussal összehasonlítva).

A didaktikai játék használatát a gyorsrendező algoritmus ismertetése előtt javasoljuk az oktatásban. Fontos, hogy a diákoknak elég idejük legyen kísérletezni a didaktikai játékkal. Ez után a játék szemléltetőeszközként való használata javasolt, amely segítségével a tanár elmagyarázza a gyorsendező algoritmust (de elmagyarázhatja más rendezési algoritmus működését is). Tapasztalataink szerint a hallgatók így könnyebben megértik a bemutatott gyorsrendezés lényeges lépéseit.

9 Láncolt lista kialakítására szolgáló interaktív animáció

Az elsős hallgatók számára az algoritmusok és a Pascal nyelv tanulása során az egyik legbonyolultabb témakör a dinamikus adatszerkezetek.

Annak érdekében, hogy a diákok jobban megértsék és el tudják képzelni a memóriában lejátszódó folyamatokat a pointer (mutató), dinamikus változó és dinamikus adatszerkezetek használata közben, készítettünk számukra egy szemléltető interaktív animációt, amely az http://anim.ide.sk/lista.php weboldalon érhető el három nyelven: magyarul, szlovákul és angolul (Stoffa & Végh, 2006b; Stoffová & Végh, 2013; Végh, 2006a).



53. ábra: Egyirányú láncolt lista kialakítása interaktív animáció segítségével

Az animáció segítségével szemléltethető a rendezetlen és rendezett egyirányú láncolt lista, ill. az egyirányú ciklikus lista létrehozásának folyamata. A diákok a dinamikus adatszerkezetek közül ezekkel találkoznak legelőször a tanulmányaik során. Úgy véljük, ha ezeken sikerül nekik

megérteni a dinamikus helyfoglalást és a mutatók használatát, később nem okoz nekik gondot bonyolultabb dinamikus adatszerkezetek létrehozása sem.

9.1 A szoftver rövid jellemzése

A felhasználó aktivitásaival irányított interaktív animáció Adobe Flash környezetben készült, ActionScript programozási nyelv használatával. Ennek köszönhetően a segédeszköz könnyen beágyazható elektronikus tananyagokba, weboldalakba.

Az animáció felhasználói felülete hat fő részből áll:

- **Pascal utasítások** – az animáció felső részében található nyomógombok, melyek segítségével a diákok kialakíthatják a láncolt listát. Bármelyik gomb megnyomásakor azonnal látszódik a memóriában történő folyamat grafikus reprezentációja.
- **Az utolsó használt utasítások** – a Pascal utasítások között, a jobb alsó részben található. Itt a diákok megtekinthetik a legutoljára használt utasításokat, melyekre itt újra rá is tudnak kattintani. Ezt felhasználva könnyen megismételhetők azok az egymás utáni utasítások, amelyek a programban egy ciklusmagban kapnak helyet.
- **Vezérlőgombok** – az utasítások alatt, jobb oldalon található kör alakú nyomógombok. Itt található a visszalépés, előre lépés, elemek átrendezése áttekinthetőbb formába, utoljára használt utasítások listájának törlése, a memóriában szemléltetett lista törlése, kimenet törlése, segítség és információk megjelenítésére szolgáló nyomógomb. Főleg az első két nyomógomb ismerete hasznos a diákoknak. Mivel ezek segítségével bármikor visszaléphetnek az előző állapotba, bátrabban kísérleteznek az animációval.
- **A lista reprezentálása a memóriában** – itt láthatók a listaelemek és azok összekapcsolása nyilakkal ábrázolt pointerek segítségével.
- **Nyelvválasztó gombok** – a lista memóriában való reprezentálása alatt, jobb oldalon találhatók az animáció nyelvének kiválasztására szolgáló nyomógombok.
- **Kimenet** – itt jelenik meg az animációban a „write” utasítás eredménye.

9.2 A diákok véleményét felmérő kérdőív kiértékelése

A 2014/15-ös akadémia év nyári szemeszterében megkértük az informatika szakos elsős hallgatókat, hogy az alkalmazás használata után értékeljék azt. A felmérésben 55 diák vett részt a Selye János Egyetem „Algoritmusok és programozás” óra keretén belül. A kérdőív teljesen anonim volt, a diákok a válaszaikat egy Google űrlap segítségével adhatták meg. A teljes kérdőív megtalálható a munka 15. mellékletében.

A hallgatók egy 5 fokozatú Likert-skála segítségével válaszolhattak a feltett kérdésekre (1 - negatív értékelés, 5 - pozitív értékelés). Az értékelések átlagait az alábbi táblázat szemlélteti.

	Mean	Std. Dev.
1. Határozd meg alkalmazás használatának HATÉKONYSÁGÁT AZ OKTATÁSBAN!	4,15	0,67
2. Segített az alkalmazás az EGYIRÁNYÚ LÁNCOLT LISTA fogalmának megértésében?	4,07	0,78
3. Mennyire segített az alkalmazás az egyirányú láncolt listát használó PASCAL PROGRAM MEGÍRÁSÁNÁL?	3,98	0,77
4. Határozd meg, mennyire FELHASZNÁLÓBARÁT, mennyire könnyen kezelhető az alkalmazás!	4,09	0,72
5. Segítettek az alkalmazásban található PIROS NYOMÓGOMBOK (PASCAL PARANCSOK) a dinamikus adatszerkezetek megértésében?	4,04	0,83
6. Segítettek az alkalmazásban az UTOLJÁRA HASZNÁLT PASCAL PARANCSOK LISTÁJA (a használat során megjelenő piros gombok a jobb oldalon) a dinamikus adatszerkezetek megértésében?	3,75	0,98
7. Segítettek az alkalmazásban a VEZÉRLŐGOMBOK (jobb oldalon található kör alakú gombok - VISSZALÉPÉS, ELŐRE LÉPÉS, LISTA ÁTRENDEZÉSE, SÚGÓ, stb.) a dinamikus adatszerkezetek megértésében?	3,95	0,92
8. Mennyire SZEMLÉLETES az alkalmazásban a láncolt lista ÁBRÁZOLÁSA a téglalapok és a nyilak segítségével?	4,27	0,72
9. Szerinted elegendő az INTERAKTIVITÁS az alkalmazásban (ahogy az alkalmazás reagál a gombnyomásokra)?	4,25	0,81
10. Javasolnád az ilyen oktatási segédeszközök használatát az oktatásban?	4,58	0,68

25. táblázat: Az animáció értékelése a diákok által (5 fokozatú Likert-skálán)

A táblázatból látható, hogy a diákok többsége pozitívan értékelte az alkalmazást, szívesen fogadnák az ilyen segédeszköz használatát az oktatásban.

9.3 Tapasztalatok és módszertani tanácsok

Úgy véljük, hogy az általunk készített segédeszköz elsősorban a tanári magyarázathoz szolgálhat hasznos szemléltetőeszközként. A diákok a segédeszköz bemutatása után saját maguk is kísérletezhetnek az alkalmazással, így egy dinamikus adatszerkezetekkel kapcsolatos Pascal program írása közben azonnal láthatják a memóriában történő folyamatokat.

Tapasztalataink szerint, ha a hallgatóknak sikerül elsajátítaniuk a rendezetlen, majd később a rendezett egyirányú láncolt lista kialakításának folyamatát az interaktív animáció segítségével, később nem okoz nekik gondot a kétirányú láncolt lista kialakítása sem.

9.4 Jövőbeli tervek, továbbfejlesztési lehetőségek

Mivel 2015 szeptemberétől a Selye János Egyetemen az informatika szakos hallgatók a Pascal programozási nyelv helyett már C nyelven tanulnak, a jövőbeli terveink közé tartozik az interaktív animáció olyan változatának létrehozása, amely a C nyelv parancsait tartalmazza.

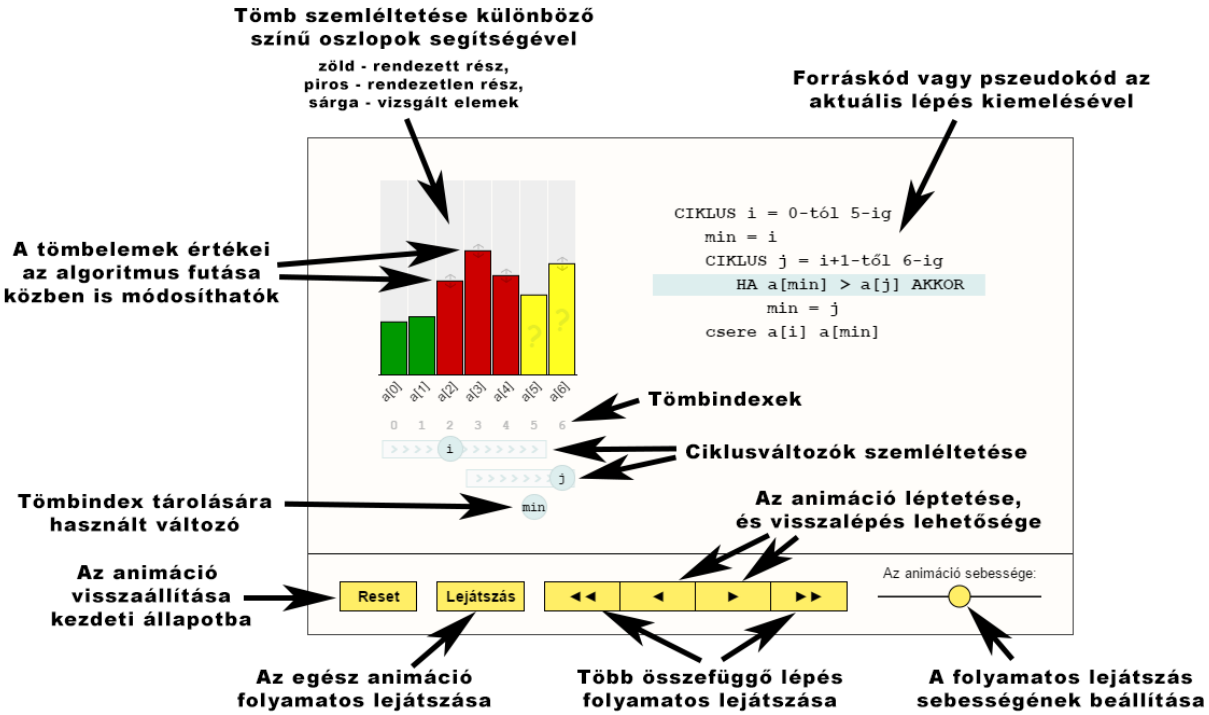
Az új változatot már HTML5 és JavaScript nyelven tervezzük megírni, hiszen az ilyen animációk weboldalon való megjelenítésére ez a technológia kezd standarddá válni a HTML5 megjelenése óta (2014 október).

10 Mikro-szintű interaktív animációk

Az első éves informatika szakos hallgatók számára a tömb használata szintén nehézségeket okozhat (Bellstrom & Thoren, 2009). Az ebben a fejezetben bemutatott, általunk fejlesztett JavaScript könyvtár olyan függvényeket tartalmaz, amely segítségégével a tananyagfejlesztők és oktatók könnyebben kialakíthatnak egydimenziós tömbökön végzett műveleteket bemutató interaktív animációkat. Az így kialakított, különböző algoritmusokat bemutató animációk külalakja, grafikus kivitelezése és vezérlése hasonló, amely megkönnyíti a diákok munkáját (Fleischer & Kucera, 2002).

10.1 Az interaktív animációk létrehozását segítő JavaScript könyvtár (inalan)

A JavaScript könyvtár elkészítésénél az objektumok kirajzolására a HTML5 Canvas elemét használtuk az objektumok kirajzolására. Ezen technika a HTML5 megjelenése óta egyre jobban kezd elterjedni és standarddá válni animációk weboldalakon való megjelenítésére. Előnye, hogy a felhasználóknak nem szükséges semmilyen beépülő modult (plugin) telepíteniük a böngészőbe.



54. ábra: Az „inalan” JavaScript könyvtárral létrehozott animáció egyes elemeinek jellemzése

Célunk olyan animációk létrehozását segítő JavaScript könyvtár megalkotása volt, amellyel viszonylag egyszerű, de lehetőleg minél több interaktivitást nyújtó animációval kísért szimulációs modellek hozhatók létre. A könyvtárral kialakítható animációk elemei és azok interaktivitása az alábbi ábrán látható (Végh, 2016c).

Az 54. ábra segítségével bemutatott interaktív elemek használatán kívül lehetőség van függvénybe való belépéskor és kilépéskor a megjelenített forráskód (pszeudokód) animálására is. Ez jól alkalmazható például rekurzív függvényhívások szemléltetésére.

A létrehozott JavaScript könyvtár, annak dokumentációja és forráskódjai az http://inalan.ide.sk/ weboldalon található hivatkozásokkal érhető el.

10.1.1 A könyvtár osztályainak bemutatása

Az interaktív animációk létrehozását segítő JavaScript könyvtár (inalan) az alábbi osztályokból áll:

- **Stage** – Minden animációnak kell egy ilyen osztályból létrehozott objektumot tartalmazni, amely a HTML5 oldal egy Canvas eleméhez kapcsolódik és oda rajzolja ki az animációt.
- **VisuVariable** – Egy oszlop segítségével szemléltetett változó.
- **VisuArray** – VisuVariable objektumokból álló tömb.
- **VisuCode** – A diákok számára megjelenítendő forráskód vagy pszeudokód.
- **VisuLabel** – Egyszerű szöveges megjegyzés, pl. rövid magyarázatok jeleníthetők meg a segítségével az animáció különböző részein.
- **VisuButton** – Nyomógomb (pl. az animáció alsó irányítópaneljén találhatók).
- **VisuScrollbar** – Görgetősáv (pl. az animáció alsó irányítópaneljén található az animáció sebességének beállítására).

Az osztályok konstruktorainak, attribútumainak és metódusainak részletes leírását a következő, angol nyelvű wiki oldalak tartalmazzák: https://github.com/veghl/inalan/wiki.

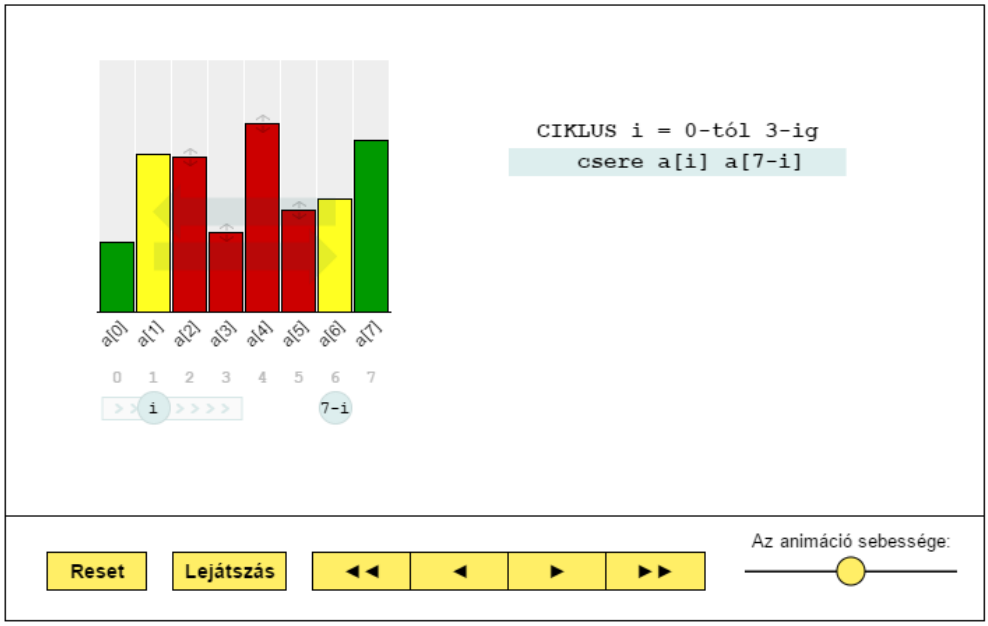
10.1.2 Animáció létrehozása a könyvtár segítségével

Egy animáció létrehozásához mindenekelőtt egy Canvas elemet tartalmazó HTML5 weboldalt kell létrehoznunk.

Ezek után létrehozhatunk egy **Stage** objektumot, melyhez a **Stage.add** metódus segítségével hozzáadhatjuk az animációban megjeleníteni kívánt elemeket (pl. VisuArray, VisuCode objektumokat).

Végül az animáció egyes lépéseit JavaScript függvényekként definiálhatjuk, majd a **Stage.setSteps** metódus segítségével megadhatjuk a lépések sorrendjét. A sorrend megadásához használhatunk utótesztelő ciklust is, melynek feltételét egy logikai értéket visszaadó JavaScript függvény képzi.

Az alábbi ábrán látható, tömb tükrözését végző animáció teljes forráskódja megtalálható a doktori értekezés 16. mellékletében.



55. ábra: Tömb tükrözését végző algoritmus animációja

10.1.3Jövőbeli tervek, továbbfejlesztési lehetőségek

A létrehozott JavaScript könyvtár segítségével jelenleg egyszerű változókon és tömb adatszerkezeteken végzett műveletek szemléltethetők. A jövőbeli terveink közé tartozik a könyvtár bővítése további adatszerkezetekkel (pl. gráf, fa).

Mivel fenn akarjuk tartani a jövőben is annak a lehetőségét, hogy a felhasználók menet közben is módosíthassák az adatokat, az animáció egyes lépéseinek megadása csak függvények segítségével oldható meg. Az egyes lépések sorrendje azonban jelenleg a függvények egymás utáni felsorolásával, vezérlésre használt változók módosításával, és utótesztelő ismétlések használatával lehetséges. Bár ilyen módon bármilyen algoritmus animációja elkészíthető, sok esetben valamivel egyszerűbb megoldás lenne további vezérlési szerkezetek használata.

Érdemes lehet tehát a jövőben a lépések sorrendjének megadását is átgondolnunk, ill. bővítenünk.

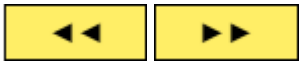
10.2 Az „inalan” JavaScript könyvtár használatával elkészített animációk

Az itt bemutatásra kerülő animációkat az előző fejezetben tárgyalt „inalan” JavaScript könyvtár segítségével készítettük el. Ennek köszönhetően az összes itt tárgyalt animáció külalakja és irányítása hasonló. Ez nagyban segítheti a diákok munkáját, hiszen elegendő egyszer elsajátítaniuk az animációk kezelését és az animációkban látottak értelmezését (Fleischer & Kucera, 2002). Az ismert környezet, biztonság- és bizalomkeltő a felhasználó számára.

A felhasználói felületen látható objektumokat és azok funkcióit részletesen szemlélteti az előző fejezet 54. ábrája.

Az oktatás során ajánlott az animációkat az alábbi három lejátszási módban egymás után használni. Amennyiben egy animációban mindhárom lejátszási mód egyidejűleg rendelkezésre áll, ezen lejátszási módokra javasolt a diákok figyelmét felhívni.

1. 

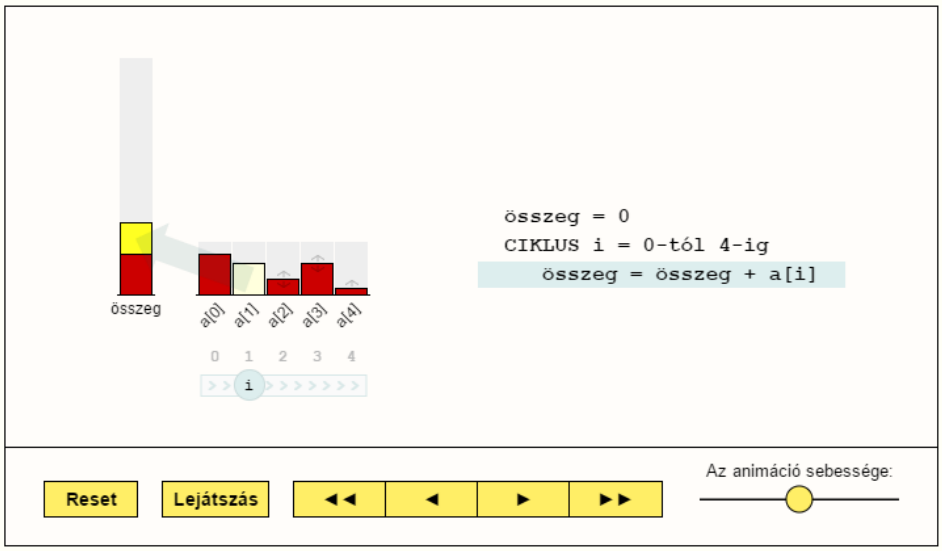
Folyamatos lejátszás – az animáció folyamatos lejátszása. A diákok az ilyen lejátszás során megfigyelhetik az algoritmus főbb lépéseit az oszlopokkal szemléltetett tömb változásának követésével. Az oszlopok színének változása és ennek információt hordozó jelentése is felismerhető.
2.  **Csak a logikailag összefüggő utasítások játszódnak le folyamatosan, ezek között a nyomógombok segítségével oldható meg a továbblépés** – itt már a diákok koncentrálhatnak a programkódra (pszeudokódra) is, ill. az oszlopokkal szemléltetett tömb alatti ciklusváltozókra.
3.  **Az algoritmus utasításonkénti léptetése** – az animáció még részletesebb, utasításonkénti lejátszására szolgál.

Az „inalan” könyvtár lehetővé teszi az egyes nyomógombok láthatatlanná tételét, és csupán a kiválasztott indexváltozók megjelenítését. Az animációk tervezői azt is eldönthetik, hogy a forráskódot meg szeretnék-e jeleníteni vagy nem. Ilyen módon a felsorolt három lejátszási

formához akár külön-külön animációk is készíthetők, melyek egymás utáni sorrendben való használatával jobban elősegíthetők az algoritmusok megértése.

10.2.1 Elemi algoritmusok animációi

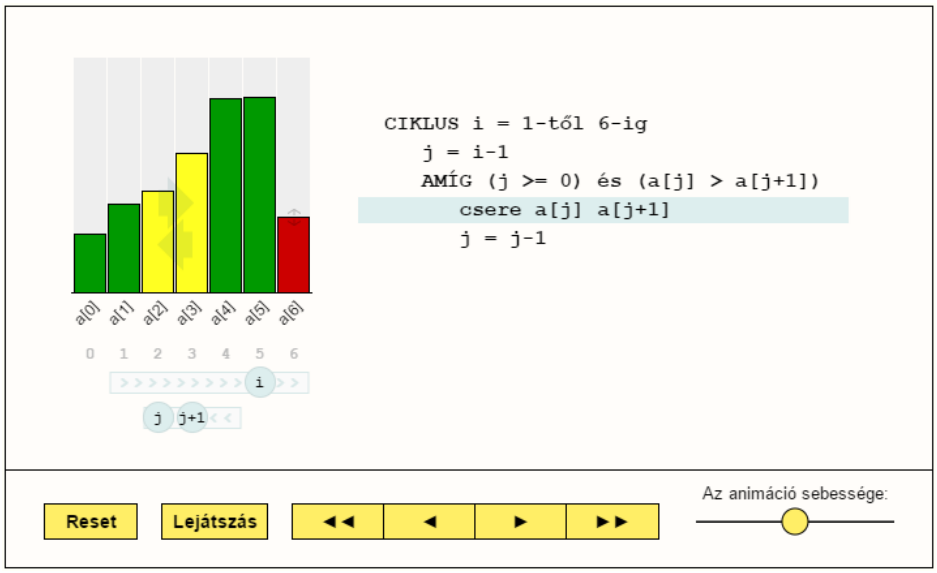
Az általunk elkészített elemi algoritmusok animációi a következő weboldalon találhatók: http://anim.ide.sk/elemi_algoritmusok.php. Ezen animációk két változó cseréjét, tömb elemeinek összegszámítását, tömb tükrözését, maximum és minimum keresését, maximum és minimum indexének keresését szemléltetik.



56. ábra: Az összegszámítást szemléltető animáció

10.2.2 Egyszerű rendezési algoritmusok animációi

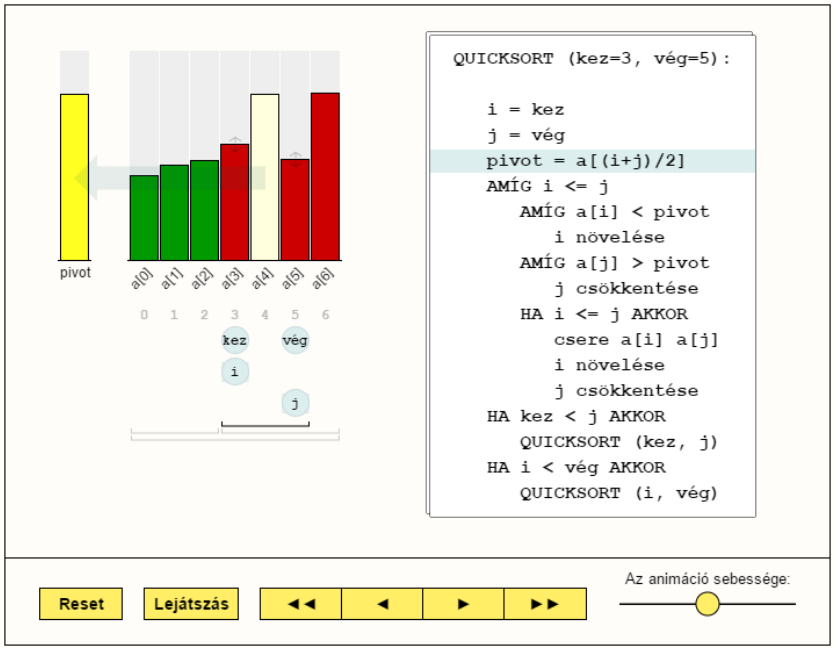
Az egyszerű rendezési algoritmusokat (egyszerű cserés rendezés, buborékrendezés, továbbfejlesztett buborékrendezés, beszúró rendezés, továbbfejlesztett beszúró rendezés, minimumkiválasztásos rendezés, maximumkiválasztásos rendezés) mutatják be az http://anim.ide.sk/rendezesi_algoritmusok_1.php weboldalon található animációk.



57. ábra: Beszúró rendezést szemléltető animáció

10.2.3 A gyorsrendezés és az összefésülő rendezés animációja

A bonyolultabb, rekurzív rendezési algoritmusok (gyorsrendezés, összefésülő rendezés) szemléltetése található az http://anim.ide.sk/rendezesi_algoritmusok_2.php weboldalon.



58. ábra: Gyorsrendezést szemléltető animáció

10.2.4 Kísérlet bemutatása

A 2015/16-os akadémia évben elvégeztünk egy pedagógiai kísérletet, melyben azt szerettük volna felmérni, hogy a diákok könnyebben megértik-e a tananyagot az általunk készített interaktív animációk segítségével, mint grafikus szemléltetés használatával. A kísérletben a

Selye János Egyetem 71 elsőéves hallgatója (négy gyakorlati csoport) vett részt a „Programozás és algoritmusok” óra keretén belül. A diákok nem voltak semmilyen szempontok alapján szétosztva, véletlenszerűen kerültek be az egyes csoportokba.

A pedagógiai kísérlet előtt a diákok kitöltöttek egy tesztet (a munka 17. melléklete), amely segítségével az eddigi ismereteiket próbáltuk meg felmérni.

A kísérlet folyamán két gyakorlati csoportban animációk segítségével (35 diáknak), a másik két gyakorlati csoportban pedig statikus grafikus szemléltetés segítségével (36 diáknak) magyaráztunk el különböző rendezési algoritmusokat: egyszerű cserés rendezést (simplesort), buborékrendezést (bubblesort), továbbfejlesztett buborékrendezést (bubblesort2), beszúró rendezést (insertsort), továbbfejlesztett beszúró rendezést (insertsort2), minimumkiválasztásos rendezést (minsort), maximumkiválasztásos rendezést (maxsort), gyorsrendezést (quicksort) és összefésülő rendezést (mergesort). Az oktatás során az interaktív animációkat használó csoportok a http://ani.ide.sk/ weboldal, míg a grafikus szemléltetést használó csoportok a http://gra.ide.sk/ weboldal segítségével ismerték meg a rendezési algoritmusokat.

Az algoritmusok azon részeit, melyek az első magyarázat után a diákoknak nem voltak világosak, részletesen is elmagyaráztunk. A magyarázat a grafikus szemléltetéseket használó csoportoknál valamivel tovább tartott, mint az animációkat használó csoportoknál. Miután a diákoknak már nem volt több kérdésük, kitöltöttek egy-egy tesztet (ezek a doktori értekezés 18-34. mellékletében találhatók).

10.2.5 Eredmények kiértékelése és elemzése

A diákok által elért pontszámok a doktori értekezés 35. mellékletében található. Mindenekelőtt megvizsgáltuk a lemért adatok normál eloszlását.

Tests of Normality

	INTERAKTÍV ANIMÁCIÓKAT használó csoportok			GRAFIKUS SZEMLÉLTETÉST használó csoportok		
	Shapiro-Wilk			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
PRETEST	0,973	34	0,539	0,918	33	0,016
SIMPLESORT	0,900	34	0,005	0,937	33	0,057
BUBBLESORT	0,791	34	0,000	0,896	33	0,004
BUBBLESORT2	0,896	31	0,006	0,959	34	0,234
INSERTSORT	0,833	31	0,000	0,868	34	0,001
INSERTSORT2	0,923	31	0,028	0,956	34	0,180
MINSORT	0,881	32	0,002	0,897	27	0,012
MAXSORT	0,822	32	0,000	0,861	27	0,002
QUICKSORT	0,880	30	0,003	0,908	23	0,036
MERGESORT	0,941	28	0,117	0,939	19	0,258

26. táblázat: A Shapio-Wilk próbák eredménye: az adatok többsége nem normál eloszlású (SPSS Statistics, 2013)

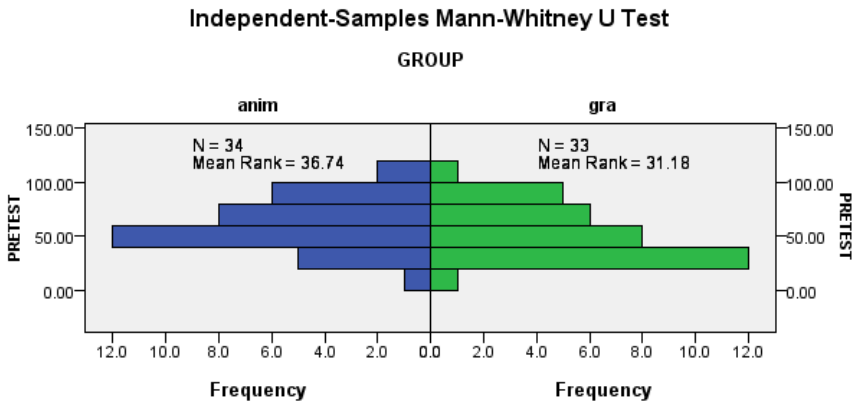
A Shapio-Wilk próbák eredményeiből látható, hogy csupán az összefésülő rendezésnél (mergesort) figyelhető meg az interaktív animációkat és a grafikus szemléltetést használó csoportok esetében is a megközelítőleg normál eloszlás. Ezért az összefésülő rendezést kivéve a csoportok mediánjait ill. rangok átlagait (mean ranks) fogjuk összehasonlítani a Mann-Whitney-Wilcoxon rangösszeg próbák (Mann-Whitney U test, SPSS Statistics, 2013) segítségével. Az összefésülő rendezés esetében viszont a csoportok átlagait hasonlítjuk majd össze független mintás t-próba (independent-samples t-test, SPSS Statistics, 2013) használatával.

Az alábbi táblázat szemlélteti az interaktív animációkat és a grafikus szemléltetést használó csoportok eredményeinek (százalékban kifejezett pontszámainak) átlagait, eloszlásait és mediánjait (Végh, 2016c).

	INTERAKTÍV ANIMÁCIÓKAT használó csoportok				GRAFIKUS SZEMLÉLTETÉST használó csoportok			
	N	Mean	Std. Dev.	Median	N	Mean	Std. Dev.	Median
PRETEST	34	59,56	23,501	58	33	53,24	24,276	50
SIMPLESORT	34	73,32	15,338	75	33	62,73	17,424	63
BUBBLESORT	34	88,09	14,154	88	33	80,55	16,216	88
BUBBLESORT2	31	77,97	16,353	75	34	52,12	25,156	50
INSERTSORT	31	87,26	12,223	88	34	81,12	13,906	88
INSERTSORT2	31	67,65	19,818	71	34	48,35	26,722	43
MINSORT	32	83,81	15,126	89	27	70,89	21,684	78
MAXSORT	32	84,94	16,679	88	27	78,89	18,352	88
QUICKSORT	30	83,77	15,110	82	23	71,35	15,683	73
MERGESORT	28	70,25	20,323	73	19	51,74	22,896	64

27. táblázat: Az elért eredmények átlagai, eloszlásai és mediánjai

Elsősorban azt próbáltuk megállapítani, hogy az előtesztelés során látható-e statisztikailag szignifikáns különbség a csoportok között.

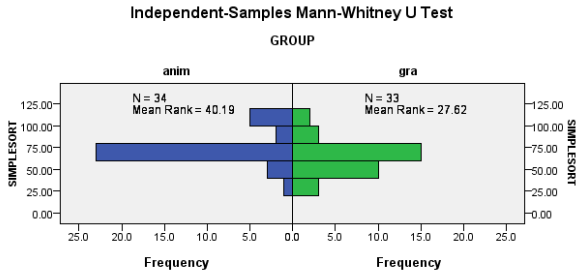


Total N	67
Mann-Whitney U	468.000
Wilcoxon W	1,029.000
Test Statistic	468.000
Standard Error	79.302
Standardized Test Statistic	-1.173
Asymptotic Sig. (2-sided test)	.241

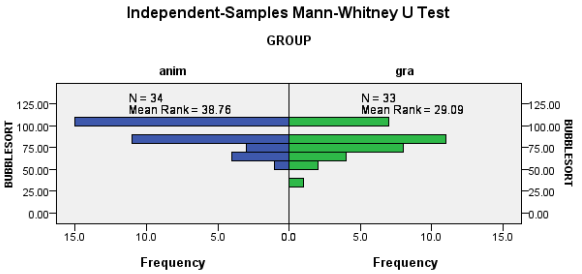
28. táblázat: Mann-Whitney-Wilcoxon rangösszeg próba eredménye (előtesztelés) (SPSS Statistics, 2013)

Az előtesztelés (pretest) során a diákok pontszámainak eloszlása hasonló volt a csoportokban. A Mann-Whitney-Wilcoxon rangösszeg próba nem mutatott ki statisztikailag szignifikáns különbséget, **U = 468, z = -1.173, p = 0.241**.

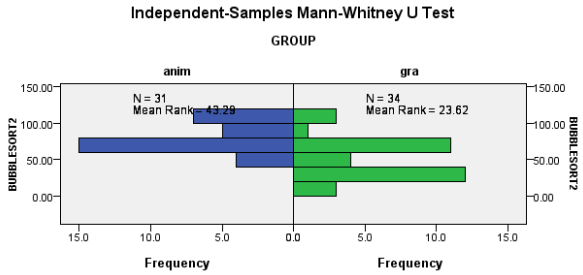
Ezek után további Mann-Whitney-Wilcoxon rangösszeg próbákat végeztünk el az egyes rendezési algoritmusok megértésének felmérésére fókuszáló tesztek eredményein.



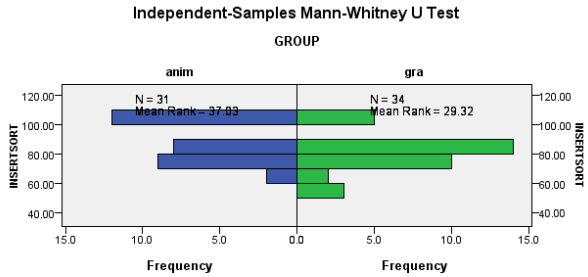
Total N	67
Mann-Whitney U	350.500
Wilcoxon W	911.500
Test Statistic	350.500
Standard Error	77.524
Standardized Test Statistic	-2.715
Asymptotic Sig. (2-sided test)	.007



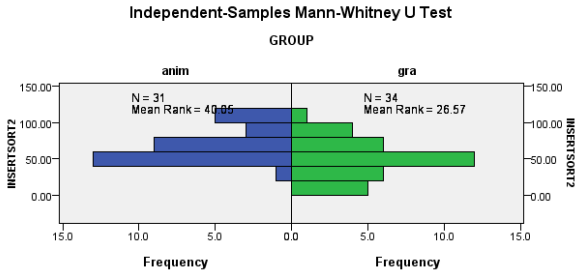
Total N	67
Mann-Whitney U	399.000
Wilcoxon W	960.000
Test Statistic	399.000
Standard Error	76.613
Standardized Test Statistic	-2.115
Asymptotic Sig. (2-sided test)	.034



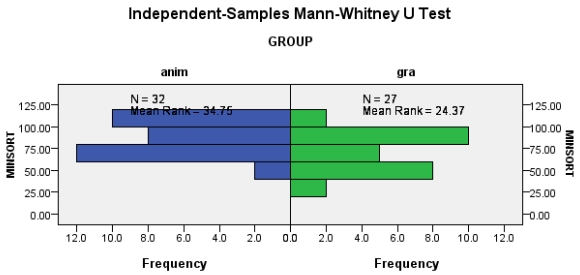
Total N	65
Mann-Whitney U	208.000
Wilcoxon W	803.000
Test Statistic	208.000
Standard Error	75.142
Standardized Test Statistic	-4.245
Asymptotic Sig. (2-sided test)	.000



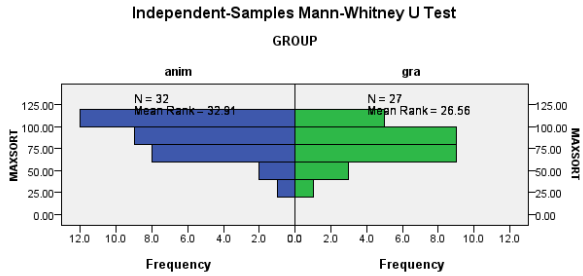
Total N	65
Mann-Whitney U	402.000
Wilcoxon W	997.000
Test Statistic	402.000
Standard Error	72.959
Standardized Test Statistic	-1.713
Asymptotic Sig. (2-sided test)	.087



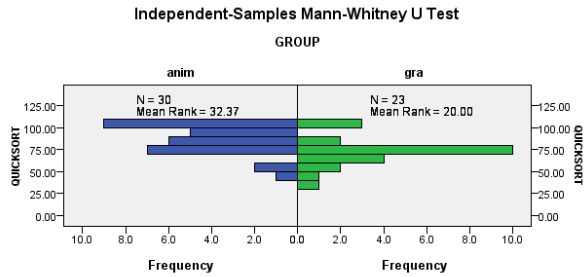
Total N	65
Mann-Whitney U	308.500
Wilcoxon W	903.500
Test Statistic	308.500
Standard Error	74.996
Standardized Test Statistic	-2.913
Asymptotic Sig. (2-sided test)	.004



Total N	59
Mann-Whitney U	280.000
Wilcoxon W	658.000
Test Statistic	280.000
Standard Error	64.266
Standardized Test Statistic	-2.365
Asymptotic Sig. (2-sided test)	.018



Total N	59
Mann-Whitney U	339.000
Wilcoxon W	717.000
Test Statistic	339.000
Standard Error	63.414
Standardized Test Statistic	-1.467
Asymptotic Sig. (2-sided test)	.142



Total N	53
Mann-Whitney U	184.000
Wilcoxon W	460.000
Test Statistic	184.000
Standard Error	54.327
Standardized Test Statistic	-2.964
Asymptotic Sig. (2-sided test)	.003

29. táblázat: Mann-Whitney-Wilcoxon rangösszeg próbák eredményei: egyszerű cserés rendezés (simplesort), buborékrendezés (bubblesort), továbbfejlesztett buborékrendezés (bubblesort2), beszúró rendezés (insertsort), továbbfejlesztett beszúró rendezés (insertsort2), minimumkiválasztásos rendezés (minsort), maximumkiválasztásos rendezés (maxsort), gyorsrendezés (quicksort) (SPSS Statistics, 2013)

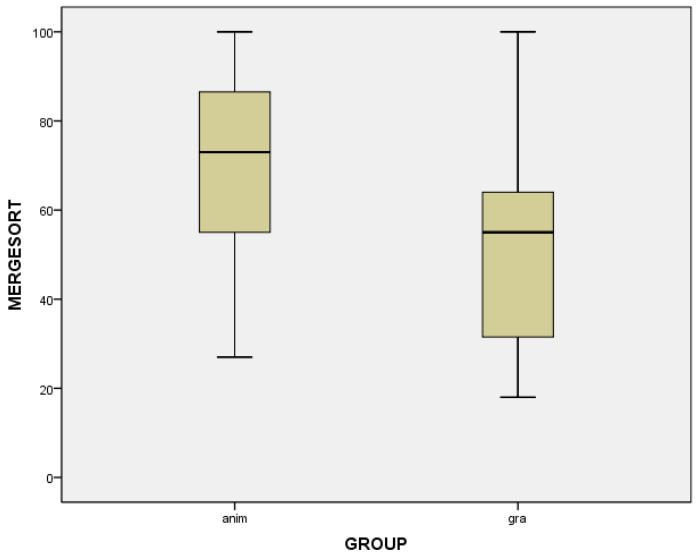
A Mann-Whitney-Wilcoxon próbák eredményeiből kiolvasható, hogy a beszúró rendezést (insertsort) és a maximumkiválasztásos rendezést leszámítva mindegyik rendezési algoritmusnál statisztikailag szignifikáns különbség észlelhető az interaktív animációkat és a grafikus szemléltetést használó csoportok között. A rangok átlagait megvizsgálva az is elmondható, hogy az interaktív animációkat használó csoportok minden esetben jobb

eredményeket értek el, mint a grafikus szemléltetést használók. A Mann-Whitney-Wilcoxon próbák eredményeit az alábbi táblázatban foglaltuk össze (Végh, 2016c).

Rendezési algoritmus	Diákok pont-számainak eloszlása hasonló a csoportok-ban?	INTERAKTÍV ANIMÁCIÓKAT használó csoportok	GRAFIKUS SZEMLÉLTETÉST használó csoportok	Eredmény
Simplesort	nem	mean rank = 40.19	mean rank = 27.62	U = 350, z = -2.715, p = 0.007
Bubblesort	nem	mean rank = 38.76	mean rank = 29.09	U = 399, z = -2.115, p = 0.034
Bubblesort2	nem	mean rank = 43.29	mean rank = 23.62	U = 208, z = -4.245, p < 0.0005
Insertsort	nem	mean rank = 37.03	mean rank = 29.32	U = 402, z = -1.713, p = 0.087
Insertsort2	nem	mean rank = 40.05	mean rank = 26.57	U = 308, z = -2.913, p = 0.004
Minsort	nem	mean rank = 34.75	mean rank = 24.37	U = 280, z = -2.365, p = 0.018
Maxsort	nem	mean rank = 32.91	mean rank = 26.56	U = 339, z = -1.467, p = 0.142
Quicksort	nem	mean rank = 32.37	mean rank = 20.00	U = 184, z = -2.964, p = 0.003

30. táblázat: Mann-Whitney-Wilcoxon rangösszeg próbák eredményeinek összefoglalása

Az összefésülő rendezés (mergesort) megértésének felmérésére fókuszáló teszt eredményein független mintás t-próbát (independent-samples t-test) végeztünk. Ennek a normálhoz hasonló eloszláson kívül egy másik feltétele, hogy ne legyenek benne ritkán előforduló kiugró értékek (outliners). Az alábbi grafikonból jól látható, hogy ez a feltétel is teljesül, tehát elvégezhető a t-próba.



59. ábra: Outlinerek vizsgálata (független mintás t-próba feltétele) (SPSS Statistics, 2013)

Group Statistics

	GROUP	N	Mean	Std. Deviation	Std. Error Mean
MERGESORT	anim	28	70,25	20,323	3,841
	gra	19	51,74	22,896	5,253

Independent Samples Test

MERGESORT	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Diff.	Std. Error Diff.	Lower	Upper
Equal variances assumed	0,384	0,539	2,912	45	0,006	18,513	6,357	5,709	31,318
Equal variances not assumed			2,845	35,607	0,007	18,513	6,507	5,311	31,715

31. táblázat: Független mintás t-próba eredménye (SPSS Statistics, 2013)

A Levene próba eredménye nem szignifikáns, tehát a szórásegyezés feltétele teljesül (p = 0.539).

A független mintás t-próba eredményéből kiderült, hogy a mergesort interaktív animációt használó diákok jobb eredményeket értek el a teszten (**70,25 ± 20,32**), mint a grafikus szemléltetést használó diákok (**51,74 ± 22,90**). A statisztikailag szignifikáns különbség **18,51 (95% CI, 5.71-től 31.32-ig), t(45) = 2.912, p = 0.006**.

Ehhez kiszámoltuk a hatásnagyságot (effect size) is:

$$s_{pooled} = \sqrt{\frac{20.323^2(28-1) + 22.896^2(19-1)}{28+19-2}} = 21.389$$

$$Cohen's\ d = \frac{|70.25 - 51.74|}{21.389} = 0.865$$

Megfigyelhető, hogy a hatásnagyság értéke erős, **d = 0.865** (> 0.8).

Mindezeket összefoglalva elmondható, hogy a kilenc rendezési algoritmus közül hétnél észlelhető szignifikáns különbség a teszteredmények összehasonlításánál, minden esetben az

interaktív animációkat használó csoportok javára. Két algoritmusnál nem mutatható ki szignifikáns különbség a csoportok között.

Ezzel sikerült bebizonyítani a 4. hipotézist (az általunk elkészített interaktív animációkat használó hallgatók jobb eredményeket érnek el a teszteken, mint a statikus grafikus szemléltetést használó társaik).

10.2.6 Tapasztalatok és módszertani tanácsok

Tapasztalataink szerint az általunk készített mikro-szintű animációk hasznos didaktikai eszközként szolgálhatnak a tömbökön végezhető műveletek és rendezési algoritmusok tanításánál és tanulásánál. A diákok a tömbelemek értékeit akár menet közben is módosíthatják, így az animáció bármelyik fázisában kipróbálhatják, hogy az algoritmus hogyan viselkedik más-más adathalmazon. Az interaktív animáció segítségével bármikor könnyedén előre- vagy visszaléphetnek az algoritmusban. Ez a lehetőség további kísérletezésekre ad alkalmat.

11 A disszertációs munka hozadéka

A munka hozadékának a következőket tartjuk:

- **Az irodalomforrásokból összegyűjtött javaslatok és elvek összegzése** – Bízunk benne, hogy ezek segítségével, mások tapasztalataira támaszkodva a jövőben sikerül több olyan interaktív animációt is létrehozni, melyek hatékonyan felhasználhatók az oktatásban.
- **A diákok tananyagformával és interaktív vizualizált szimulációs modellekkel támogatott tanulással kapcsolatos véleményének felmérése** – A kérdőívben a diákok több mint 80%-a az animációt részesítette előnyben más tananyagformákkal szemben az algoritmusok tanulásánál. Az animáció használatával tehát érdemes foglalkozni az oktatásban, hiszen növeli a tananyag szemléltetésének fokát, támogatja az aktív tanulást, s többek között nagy motiváló erővel is bír.
- **A Second Life virtuális világban kialakított tér** – Az általunk kialakított „Luckstone suli" virtuális tere bárki számára hozzáférhető. Reméljük, hogy a technika és a virtuális világ rohamos fejlődésével az általunk említett hátrányok eltörpülnek, és a jövőben a virtuális világ előnyeit felhasználva szélesebb körben is teret hódít az informatikaoktatásban.
- **Algoritmusok animációinak gyűjteménye (http://algoanim.ide.sk)** – Az általunk megalkotott portálon megpróbáltuk összegyűjteni az oktatásban használható animációkat, melyeket a jövőben folyamatosan fogunk bővíteni továbbiakkal. Bízunk benne, hogy egy ilyen portál nagy segítséget nyújt a diákoknak az informatikai algoritmusokkal való ismerkedésük során. A tanári informatika szakos hallgatók esetében a tanulási stílus tanítási stílussá válhat.
- **Weboldalba beágyazható interaktív animációk (http://anim.ide.sk)** – Az általunk elkészített interaktív animációk hasznos segédeszközként szolgálhatnak a rendezési algoritmusok és a dinamikus adatszerkezetek oktatásánál. A tanári informatika szakos hallgatók esetében saját didaktikai vizualizált szimulációs modellek fejlesztéséhez is vezethet.
    - **Interaktív rendezés kártyalapok segítségével** – Az animációk segítségével a diákok megismerhetik az egyszerű rendezési algoritmusok

lényeges lépéseit és a köztük levő különbségeket. Az elvégzett pedagógiai kísérlet eredményei is ezt támasztják alá.

- **Didaktikai játék: Ládák rendezése minimális összehasonlítással** – A didaktikai játék segítségével sikerült a diákoknak lecsökkenteniük az redundáns mérések (összehasonlítások) számát és így „megalkotni" egy saját rendezési algoritmust. Ezt pedagógiai kísérlettel is bebizonyítottuk, mely alapja lehet további kutatásoknak. A játék kiváló segédeszközként szolgálhat a gyorsrendezés megértéséhez, valamint a rendezési algoritmusok közti hatékonyság vizsgálatához is.
- **Láncolt lista kialakítására szolgáló interaktív animáció** – A láncolt lista kialakítására használható animáció segítségével a diákok könnyebben el tudják képzelni a memóriában lejátszódó folyamatokat a kiválasztott utasítások elvégzése alatt.

- **Interaktív animációk létrehozását segítő JavaScript könyvtár (inalan) –** Az általunk létrehozott JavaScript könyvtár segítségével könnyebben létrehozhatók interaktív, weboldalba beágyazható animációk.
  - **Az „inalan" JavaScript könyvtár használatával elkészített animációk** – A kialakított animációk (tömbműveletek, rendezési algoritmusok) segítségével a diákok gyorsabban megértik a bemutatott algoritmusokat, spontánul tanulnak, és az ismeret és jártasság szerzését nem veszik terhelésnek. A pedagógiai kísérlet azt is bebizonyította, hogy az ilyen interaktív animációkat használó diákok jobb eredményeket értek el a teszteken, mint a statikus grafikus szemléltetést használó társaik.

12 Befejezés

A disszertációs munkában bemutatott interaktív animációk és a hozzájuk tartozó pedagógiai kísérletek eredményei azt támasztják alá, hogy az animációknak és a vizualizációknak fontos szerepük van az informatika oktatásában. Algortimus-animációk segítségével az elsős informatika szakos hallgatók könnyebben, rövidebb idő alatt, sokszor izgalmas, játékos formában érthetik meg és sajátíthatják el a szemléltetett fogalmakat, folyamatokat és az azok közti különbségeket.

Fontos azonban, hogy egy új animáció megtervezésénél, ill. fejlesztésénél figyelembe vegyük a doktori értekezésben összefoglalt multimédia elveket és javaslatokat, amelyek az optimális tanulás és új ismeretszerzés folyamatát biztosítják. Ehhez szükséges az is, hogy megválasszuk a megfelelő grafikus reprezentációt a módszertani elveket betartva, és nem utolsó sorban, hogy megfelelő interaktivitást iktassunk az animációkba, amely lehetőséget ad az animált szimulációs kísérletek irányítására. Az ilyen animációval kísért szimulációs kísérlet aktív tanuláshoz vezet a konstruktivizmus értelmében, továbbá a Bloom taxonómia alapján a tanulás és ismeretszerzés legmagasabb fokának elérését segíti elő. A gondosan megtervezett, implementált és tesztelt interaktív animációk hasznos oktatási segédeszközként szolgálhatnak úgy a tanárok, mint a diákok számára.

13 Irodalomjegyzék

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., . . . Wittrock, M. C. (2001). *A Taxonomy for Learning, Teaching, and Assessing. A Revision of Bloom's Taxonomy of Educational Objectives*. USA: Addison Wesley Longman, Inc.

Bellstrom, P., & Thoren, C. (2009). Learning How to Program through Visualization: A Pilot Study on the Bubble Sort Algorithm. *2009 Second International Conference on the Applications of Digital Information and Web Technologies (Icadiwt 2009)*, 90-94. doi:10.1109/icadiwt.2009.5273943

Bernát, P. (2014). The Methods And Goals Of Teaching Sorting Algorithms In Public Education. *Acta Didactica Napocensia, 7*(2), 10.

Bloom, B. S., Englehart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). *The Taxonomy of Educational Objectives, The Classification of Educational Goals, Handbook I: Cognitive Domain* (B. S. Bloom Ed.). New York: David McKay Company, Inc.

Byrne, M. D., Catrambone, R., & Stasko, J. T. (1999). Evaluating animations as student aids in learning computer algorithms. *Computers & Education, 33*(4), 253-278. doi:10.1016/s0360-1315(99)00023-8

Csízi, L., & Végh, L. (2011). *Virtuális világok az oktatásban.* Paper presented at the III. Oktatás-Informatikai Konferencia, Budapest.

Esponda-Arguero, M. (2010). Techniques for visualizing data structures in algorithmic animations. *Information Visualization, 9*(1), 31-46.

Esteves, M., Fonseca, B., Morgado, L., & Martins, P. (2009). Using Second Life for Problem Based Learning in Computer Science Programming. *Journal of Virtual Worlds Research, 2*(1).

Esteves, M., Fonseca, B., Morgado, L., & Martins, P. (2011). Improving teaching and learning of computer programming through the use of the Second Life virtual world. *British Journal of Educational Technology, 42*(4), 624-637. doi:10.1111/j.1467-8535.2010.01056.x

Fleischer, R., & Kucera, L. (2002). Algorithm animation for teaching. *Software Visualization, 2269*, 113-128.

Furcy, D., Naps, T., & Wentworth, J. (2008). Sorting Out Sorting - The Sequel. *Iticse '08: Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, 174-178.

Geigel, J. (2009). *Teaching animation in Second Life*. Paper presented at the SIGGRAPH 2009: Talks, New Orleans, Louisiana.

Geigel, J. (2010). Teaching Animation in Virtual Space: The Use of Second Life as An Extended Approach for Teaching Computer Graphics Course On-Line. *The Journal of Virtual Worlds and Education, 1*(1), 37-62.

Grissom, S., McNally, M. F., & Naps, T. (2003). *Algorithm visualization in CS education: comparing levels of student engagement.* Paper presented at the Proceedings of the 2003 ACM symposium on Software visualization, San Diego, California.

Gubo, Š., Jaruska, L., & Végh, L. (2012). *The Possibilities of Using Virtual Worlds in Teaching Basics of 3D Modelling and Simulation (Možnosti využitia virtuálnych svetov vo vyučovaní základov 3D modelovania a simulácie).* Paper presented at the XXX. International Colloquium on the Management of Educational Process, Brno, CZ.

Hansen, S., Narayanan, N. H., & Hegarty, M. (2002). Designing educationally effective algorithm visualizations. *Journal of Visual Languages and Computing, 13*(3), 291-317. doi:10.1006/s1045-926x(02)00027-7

Hundhausen, C., & Douglas, S. (2000). Using visualizations to learn algorithms: Should students construct their own, or view an expert's? *2000 Ieee International Symposium on Visual Languages, Proceedings*, 21-28.

Hundhausen, C. D., & Brown, J. L. (2008). Designing, visualizing, and discussing algorithms within a CS 1 studio experience: An empirical study. *Computers & Education, 50*(1), 301-326. doi:10.1016/j.compedu.2006.06.002

Hundhausen, C. D., Douglas, S. A., & Stasko, J. T. (2002). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing, 13*(3), 259-290. doi:10.1006/s1045-926x(02)00028-9

Inman, C., Wright, V. H., & Hartman, J. A. (2010). Use Second Life in K-12 and Higher Education: A review of Research. *Journal of Interactive Online Learning, 9*(1).

Kann, C., Lindeman, R. W., & Heller, R. (1997). Integrating algorithm animation into a learning environment. *Computers & Education, 28*(4), 223-228. doi:10.1016/s0360-1315(97)00015-8

Katai, Z., & Tóth, L. (2010). Technologically and artistically enhanced multi-sensory computer-programming education. *Teaching and Teacher Education, 26*(2), 244-251. doi:http://dx.doi.org/10.1016/j.tate.2009.04.012

Kehoe, C., Stasko, J., & Taylor, A. (2001). Rethinking the evaluation of algorithm animations as learning aids: an observational study. *International Journal of Human-Computer Studies, 54*(2), 265-284. doi:10.1006/ijhc.2000.0409

Kiss, P. (2014). *Angol nyelv oktatásának lehetőségei virtuális világokban.* (Bc.), Selye János Egyetem, Komárno, SK.

Kristóf, Z., Végh, L., & Bodnár, K. (2011). Felsőoktatásban alkalmazott Sloodle eszközrendszer használati tapasztalatai. Egy saját eszköz bemutatása. *Oktatás-Informatika, 2*(3-4), 68-76.

Kuk, K., Jovanovic, D., Jokanovic, D., Spalevic, P., Caric, M., & Panic, S. (2012). Using a game-based learning model as a new teaching strategy for computer engineering. *Turkish Journal of Electrical Engineering and Computer Sciences, 20*, 1312-1331. doi:10.3906/elk-1101-962

Lattu, M., Meisalo, V., & Tarhio, J. (2003). A visualisation tool as a demonstration aid. *Computers & Education, 41*(2), 133-148. doi:10.1016/s0360-1315(03)00032-0

Marešová, H. (2010). Vzdělávaní v Second Life (Education in Second Life). *Nové technológie ve vzdělávání*. Retrieved from http://www.kteiv.upol.cz/ntvv/ntvv_2010_sbornik.pdf

Mayer, R. E. (2009). *Multimedia Learning* (second ed.). New York, USA: Cambridge University Press.

Michels, P. (2010). Universities Use Second Life to Teach Complex Concepts. Retrieved from http://www.govtech.com/education/universities-use-second-life-to-teach.html

Naps, T., & Grissom, S. (2002). The effective use of quicksort visualizations in the classroom. *J. Comput. Sci. Coll., 18*(1), 88-96.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., . . . Velázquez-Iturbide, J. Á. (2002). Exploring the role of visualization and engagement in computer science education. *SIGCSE Bull., 35*(2), 131-152. doi:10.1145/782941.782998

Nádasi, A. Oktatáselmélet és technológia. Retrieved from http://okt.ektf.hu/data/nadasia/file/tananyag/oktataselmelet/1_tananyag13.html

Palmiter, S., & Elkerton, J. (1991). *An evaluation of animated demonstrations of learning computer-based tasks*. Paper presented at the Human Factors in Computing Systems: Proceedings of the SIGCHI Conference, (CHI '91).

Patwardhan, M., & Murthy, S. (2015). When does higher degree of interaction lead to higher learning in visualizations? Exploring the role of 'Interactivity Enriching Features'. *Computers & Education, 82*, 292-305. doi:10.1016/j.compedu.2014.11.018

Penfold, P. (2008). Learning Through the World of Second Life - A Hospitality and Tourism Experience. *Journal of Teaching in Travel & Tourism, 8*(2 & 3), 139-160.

Pereira, A., Martins, P., Morgado, L., & Fonseca, B. (2009). A virtual environment study in entrepreneurship education of young children. *Journal of Virtual Worlds Research: Pedagogy, Education and Innovation in 3-D Virtual Worlds*(April, 2009).

Roush, P., Nie, M., & Wheeler, M. (2009). Between Snapshots and Avatars: Using Virtual Methodologies for Fieldwork in Second Life. *Journal of Virtual Research: Pedagogy, Education and Innovation in 3-D Virtual Worlds*(April, 2009).

Rudder, A., Bernard, M., & Mohammed, S. (2007). Teaching programming using visualization. *Proceedings of the Sixth IASTED International Conference on Web-Based Education*, 487-492.

Smith, M., & Berge, Z. L. (2009). Social Learning Theory in Second Life. *MERLOT Journal of Online Learning and Teaching, 5*(2).

Sommerville, I. (2007). *Szoftverrendszerek fejlesztése* (8. ed.). Budapest: Panem Könyvkiadó.

Stoffa, V. (2003). *Computer-aided learning of programming*. Paper presented at the Proceedings of the 4th international conference conference on Computer systems and technologies: e-Learning, Rousse, Bulgaria.

Stoffa, V., & Végh, L. (2006a). A programozás tanításának és tanulásának elektronikus támogatása. *Eruditio-Educatio, 3 (2006)*, 105-113.

Stoffa, V., & Végh, L. (2006b). *Guided animation of dynamic data structures*. Paper presented at the Third Central European Multimedia and Virtual Reality Conference, Eger, Hungary.

Stoffa, V., & Végh, L. (2014). *Didaktikai kutatásra szolgáló adatbegyűjtő információs rendszer*. Paper presented at the Agria Media 2014, Eger, HU.

Stoffová, V. (2004). *Počítač – univerzálny didaktický prostriedok* (1. ed.). Nitra, Slovakia: Fakulta prírodných vied UKF v Nitre.

Stoffová, V. (2005a). *Animation Models in E-learning of Science.* Paper presented at the Integrating New Technologies in Science and Education, Žilina, Slovakia.

Stoffová, V. (2005b). *Controlled simulation and animation in computer presentations.* Paper presented at the International Conference on Computer Systems and Technologies - CompSysTech 2005, Varna, Bulgaria.

Stoffová, V., & Czakóová, K. (2016). *Úvod do programovania v prostredí mikrosvetov*. Komárno, Slovakia: J. Selye University.

Stoffová, V., & Végh, L. (2007). *Tvorba animačno-simulačných modelov v rôznych prostrediach*. Paper presented at the Informatika v škole a v praxi, Ružomberok, Slovakia.

Stoffová, V., & Végh, L. (2010). *Szemléltető animációk a programozásban.* Paper presented at the INFODIDACT 2010, Szombathely.

Stoffová, V., & Végh, L. (2013). *Grafické modely dynamických údajových štruktúr a ich význam vo vyučovaní programovania (Visual models of dynamic data structures and their importance in teaching programming).* Paper presented at the Trendy ve vzdělávání 2013, Olomouc, CZ.

Stoffová, V., & Végh, L. (2014). *Prieskum vhodnosti používania animačno-simulačných modelov algoritmov vo vzdelávaní (Survey of Effectiveness of Animation-Simulation Models of Algorithms in Education).* Paper presented at the XXVII. DIDMATTECH 2014, Olomouc, CZ.

Sülei, C. (2013). *3D modellek készítésének oktatási lehetőségei a virtuális világokban.* (Bc.), Selye János Egyetem, Komárno, SK.

Turcsányi-Szabó, M., Csízi, L., & Végh, L. (2012). Virtual Worlds in Education - best practice, design and research consideration. *Teaching Mathematics and Computer Science, 10*(2), 309-323. doi:10.5485/TMCS.2012.0308

Urquiza-Fuentes, J., & Velazquez-Iturbide, J. A. (2013). Toward the effective use of educational program animations: The roles of student's engagement and topic complexity. *Computers & Education, 67*, 178-192. doi:10.1016/j.compedu.2013.02.013

Végh, L. (2006a). *Elektronická podpora vyučovania dynamických údajových štruktúr (Electronic support of teaching dynamic data structures)*. Paper presented at the XIX. DIDMATTECH 2006, Komárno.

Végh, L. (2006b). *Vizualizácia algoritmov vo vyučovaní programovania*. Paper presented at the Informatika v škole a v praxi, Ružomberok, Slovakia.

Végh, L. (2010a). *Tvorba animácií pre e-learning (Creating animations for e-learning)*. Paper presented at the XXII. DIDMATTECH 2009, Trnava, Slovakia.

Végh, L. (2010b). *Vyučovanie algoritmizácie a programovania hravou formou (Teaching algorithmization and programming plazfully)*. Paper presented at the XXVIII International Colloquium on the Management of Educational Process, Brno, Czech Republic.

Végh, L. (2011a). *Animations in Teaching Algorithms and Programming (Animácie vo vyučovaní algoritmov a programovania).* Paper presented at the Nové technologie ve vzdělávání, Olomouc, CZ.

Végh, L. (2011b). *From Bubblesort to Quicksort with Playing a Game (Hravou formou od bublinkového triedenia po rýchle triedenie).* Paper presented at the XXIX. International Colloquium on the Management of Educational Process, Brno, CZ.

Végh, L. (2012). *Informatikai tantárgyak oktatásának lehetőségei a virtuális világokban.* Paper presented at the IV. Oktatás-Informatikai Konferencia, Budapest.

Végh, L. (2014a). *Methods of Creating Educational 3D Animations Models in Virtual Worlds* Paper presented at the XXVI. DIDMATTECH 2013, Győr.

Végh, L. (2014b). *Programozás alapjainak oktatása az LSL szkriptnyelv használatával.* Paper presented at the Oktatás és tudomány a XXI. század elején, Komárno.

Végh, L. (2016a). *Interaktív algoritmus animációk az oktatásban.* Paper presented at the XXIX. DIDMATTECH 2016, Budapest, HU.

Végh, L. (2016b). Interaktívne animácie vo vyučovaní algoritmov (Interactive animations in teaching and learning programming). *Edukacja – Technika – Informatyka (Education – Technology – Computer Science), 15*(1), 207-211. doi:10.15584/eti.2016.1.29

Végh, L. (2016c). Javascript library for developing interactive micro-level animations for teaching and learning algorithms on one-dimensional arrays. *Acta Didactica Napocensia, 9*(2), 23-32.

Végh, L. (2016d). *Second Life virtuális világban kialakított oktatási környezet felhasználási lehetőségei.* Paper presented at the SJE Nemzetközi Tudományos Konferencia 2016, Komárno, Slovakia.

Végh, L. (2016e). Using Interactive Game-based Animations for Teaching and Learning Sorting Algorithms. *eLearning and Software for Education, 1*(2016), 565-570. doi:10.12753/2066-026X-16-083

Végh, L., & Csízi, L. (2010). *Using Virtual Worlds in Education (Využitie virtuálnych svetov vo vzdelávaní).* Paper presented at the Education and Technology / Edukacja i Technika, Radom, PL.

Végh, L., Gubo, Š., & Takáč, O. (2015). *Interactive Algorithm Animations in Virtual Worlds (Interaktívne animácie algoritmov vo virtuálnych svetoch).* Paper presented at the XXXIII. International Colloquium on the Management of Educational Process, Brno, CZ.

Végh, L., & Stoffová, V. (2016). An interactive animation for learning sorting algorithms: How students reruced the number of comparisons in a sorting algorithm by playing a didactic game. *Teaching Mathematics and Computer Science, 14*(1), 45-62. doi:10.5485/TMCS.2016.0415

Végh, L., & Stoffová, V. (2017). Algorithm Animations for Teaching and Learning the Main Ideas of Basic Sortings. *Informatics in Education, 16*(1), 121-140. doi:10.15388/infedu.2017.07

Végh, L., & Takáč, O. (2017) Using interactive card animations for understanding of the essential aspects of non-recursive sorting algorithms. In*: Vol. 511 AISC. Federated Conference on Software Development and Object Technologies, SDOT 2015* (pp. 336-347): Springer Verlag.

Végh, L., & Turcsányi-Szabó, M. (2017). Using a Virtual School for Teaching and Learning the Basics of 3D Modeling and LSL Scripting in Second Life. In *Could technology support learning efficiency?, Vol I* (pp. 572–579). Bucharest: Carol I Natl Defence Univ Publishing House.

Wang, F. H., & Burton, J. K. (2013). Second Life in education: A review of publications from its launch to 2011. *British Journal of Educational Technology, 44*(3), 357-371. doi:10.1111/j.1467-8535.2012.01334.x

Young, J. R. (2002). Homework? What Homework? Students Seem to Be Spending Less Time Studying Than They Used To. *The Chronicle of Higher Education, 49*(15), A35-A37.

14 A szerző disszertációhoz kapcsolódó publikációi

Csízi, L., & Végh, L. (2011). *Virtuális világok az oktatásban.* Paper presented at the III. Oktatás-Informatikai Konferencia, Budapest.

Gubo, Š., Jaruska, L., & Végh, L. (2012). *The Possibilities of Using Virtual Worlds in Teaching Basics of 3D Modelling and Simulation (Možnosti využitia virtuálnych svetov vo vyučovaní základov 3D modelovania a simulácie).* Paper presented at the XXX. International Colloquium on the Management of Educational Process, Brno, CZ.

Kristóf, Z., Végh, L., & Bodnár, K. (2011). Felsőoktatásban alkalmazott Sloodle eszközrendszer használati tapasztalatai. Egy saját eszköz bemutatása. *Oktatás-Informatika, 2*(3-4), 68-76.

Stoffa, V., & Végh, L. (2006a). A programozás tanításának és tanulásának elektronikus támogatása. *Eruditio-Educatio, 3 (2006)*, 105-113.

Stoffa, V., & Végh, L. (2006b). *Guided animation of dynamic data structures*. Paper presented at the Third Central European Multimedia and Virtual Reality Conference, Eger, Hungary.

Stoffa, V., & Végh, L. (2014). *Didaktikai kutatásra szolgáló adatbegyűjtő információs rendszer*. Paper presented at the Agria Media 2014, Eger, HU.

Stoffová, V., & Végh, L. (2007). *Tvorba animačno-simulačných modelov v rôznych prostrediach*. Paper presented at the Informatika v škole a v praxi, Ružomberok, Slovakia.

Stoffová, V., & Végh, L. (2010). *Szemléltető animációk a programozásban.* Paper presented at the INFODIDACT 2010, Szombathely.

Stoffová, V., & Végh, L. (2013). *Grafické modely dynamických údajových štruktúr a ich význam vo vyučovaní programovania (Visual models of dynamic data structures and their importance in teaching programming).* Paper presented at the Trendy ve vzdělávání 2013, Olomouc, CZ.

Stoffová, V., & Végh, L. (2014). *Prieskum vhodnosti používania animačno-simulačných modelov algoritmov vo vzdelávaní (Survey of Effectiveness of Animation-Simulation Models of Algorithms in Education).* Paper presented at the XXVII. DIDMATTECH 2014, Olomouc, CZ.

Turcsányi-Szabó, M., Csízi, L., & Végh, L. (2012). Virtual Worlds in Education - best practice, design and research consideration. *Teaching Mathematics and Computer Science, 10*(2), 309-323. doi:10.5485/TMCS.2012.0308

Végh, L. (2006a). *Elektronická podpora vyučovania dynamických údajových štruktúr (Electronic support of teaching dynamic data structures)*. Paper presented at the XIX. DIDMATTECH 2006, Komárno.

Végh, L. (2006b). *Vizualizácia algoritmov vo vyučovaní programovania*. Paper presented at the Informatika v škole a v praxi, Ružomberok, Slovakia.

Végh, L. (2010a). *Tvorba animácií pre e-learning (Creating animations for e-learning).* Paper presented at the XXII. DIDMATTECH 2009, Trnava, Slovakia.

Végh, L. (2010b). *Vyučovanie algoritmizácie a programovania hravou formou (Teaching algorithmization and programming plazfully).* Paper presented at the XXVIII International Colloquium on the Management of Educational Process, Brno, Czech Republic.

Végh, L. (2011a). *Animations in Teaching Algorithms and Programming (Animácie vo vyučovaní algoritmov a programovania).* Paper presented at the Nové technologie ve vzdělávání, Olomouc, CZ.

Végh, L. (2011b). *From Bubblesort to Quicksort with Playing a Game (Hravou formou od bublinkového triedenia po rýchle triedenie).* Paper presented at the XXIX. International Colloquium on the Management of Educational Process, Brno, CZ.
Végh, L. (2012). *Informatikai tantárgyak oktatásának lehetőségei a virtuális világokban.* Paper presented at the IV. Oktatás-Informatikai Konferencia, Budapest.
Végh, L. (2014a). *Methods of Creating Educational 3D Animations Models in Virtual Worlds* Paper presented at the XXVI. DIDMATTECH 2013, Győr.
Végh, L. (2014b). *Programozás alapjainak oktatása az LSL szkriptnyelv használatával.* Paper presented at the Oktatás és tudomány a XXI. század elején, Komárno.
Végh, L. (2016a). *Interaktív algoritmus animációk az oktatásban.* Paper presented at the XXIX. DIDMATTECH 2016, Budapest, HU.
Végh, L. (2016b). Interaktívne animácie vo vyučovaní algoritmov (Interactive animations in teaching and learning programming). *Edukacja – Technika – Informatyka (Education – Technology – Computer Science), 15*(1), 207-211. doi:10.15584/eti.2016.1.29
Végh, L. (2016c). Javascript library for developing interactive micro-level animations for teaching and learning algorithms on one-dimensional arrays. *Acta Didactica Napocensia, 9*(2), 23-32.
Végh, L. (2016d). *Second Life virtuális világban kialakított oktatási környezet felhasználási lehetőségei.* Paper presented at the SJE Nemzetközi Tudományos Konferencia 2016, Komárno, Slovakia.
Végh, L. (2016e). Using Interactive Game-based Animations for Teaching and Learning Sorting Algorithms. *eLearning and Software for Education, 1*(2016), 565-570. doi:10.12753/2066-026X-16-083
Végh, L., & Csízi, L. (2010). *Using Virtual Worlds in Education (Využitie virtuálnych svetov vo vzdelávaní).* Paper presented at the Education and Technology / Edukacja i Technika, Radom, PL.
Végh, L., Gubo, Š., & Takáč, O. (2015). *Interactive Algorithm Animations in Virtual Worlds (Interaktívne animácie algoritmov vo virtuálnych svetoch).* Paper presented at the XXXIII. International Colloquium on the Management of Educational Process, Brno, CZ.
Végh, L., & Stoffová, V. (2016). An interactive animation for learning sorting algorithms: How students reruced the number of comparisons in a sorting algorithm by playing a didactic game. *Teaching Mathematics and Computer Science, 14*(1), 45-62. doi:10.5485/TMCS.2016.0415
Végh, L., & Stoffová, V. (2017). Algorithm Animations for Teaching and Learning the Main Ideas of Basic Sortings. *Informatics in Education, 16*(1), 121-140. doi:10.15388/infedu.2017.07
Végh, L., & Takáč, O. (2017) Using interactive card animations for understanding of the essential aspects of non-recursive sorting algorithms. In*: Vol. 511 AISC. Federated Conference on Software Development and Object Technologies, SDOT 2015* (pp. 336-347): Springer Verlag.
Végh, L., & Turcsányi-Szabó, M. (2017). Using a Virtual School for Teaching and Learning the Basics of 3D Modeling and LSL Scripting in Second Life. In *Could technology support learning efficiency?, Vol I* (pp. 572–579). Bucharest: Carol I Natl Defence Univ Publishing House.

Összefoglalás

A doktori értekezés az informatikai algoritmusok oktatásában hatékonyan felhasználható interaktív animációinak tervezésére és használatára fókuszál. A munkában mindenekelőtt a rendezési algoritmusok animációra összpontosítottunk, azonban az összegyűjtött javaslatok, tapasztalatok más algoritmus-animációk tervezésénél, implementálásánál, és a tanítási-tanulási folyamatban való alkalmazásánál is hasznosak lehetnek.

Az utóbbi 30-35 évben végzett, különféle publikációkban megjelent pedagógiai kísérletek eredményei nem minden esetben mutatták ki az animációk hatékonyságát az oktatásban. A doktori értekezés első részében összegyűjtöttük azokat az elveket és javaslatokat, amelyek alkalmazásával hatékony oktatási segédeszközök – interaktív animációk – fejleszthetők. Ezek alapján kísérletet tettünk a virtuális világokban történő oktatásra, majd elkészítettünk egy olyan saját fejlesztésű háromnyelvű portált, amely interneten fellelhető algoritmus-animációk gyűjteményét tartalmazza.

A továbbiakban saját algoritmus-animációkat készítettünk Adobe Flash, ill. HTML5/JavaScript technológiák felhasználásával, mindenekelőtt a rendezési algoritmusokra fókuszálva. Az általunk elkészített animációk két csoportra oszthatók. Az első csoportba tartoznak azok az animációk, amelyek a rendezési algoritmusok lényeges lépéseinek megismerésére és az egyes rendezési algoritmusok közötti lényeges különbségek felismerésére összpontosítanak (ilyenek a faládák és kártyalapok rendezésére szolgáló interaktív animációk – didaktikai játékok formájában). A második csoportba azok az animációk tartoznak, melyek már részletesen mutatják be az egyes algoritmusokat (ilyen a dinamikus láncolt lista kialakításának szemléltetésére használható, ill. a tömbön végezhető egyszerű algoritmusokat és a rendezési algoritmusokat bemutató interaktív animációk). Az utóbbi animációk elkészítésének egyszerűsítéséhez egy JavaScript könyvtárat is megkíséreltünk létrehozni („inalan” JavaScript könyvtár).

A saját fejlesztésű animációk segítségével több pedagógiai kísérletet is végrehajtottunk a 2014/15 és 2015/16-os akadémia évben, az elsős informatika szakos egyetemi hallgatók körében. Az összegyűjtött adatokat különféle statisztikai módszerek alkalmazásával, az SPSS Statistics szoftver segítségével értékeltük ki. A kísérletek többsége olyan pozitív eredményeket mutatott ki, melyek alátámasztják az általunk létrehozott interaktív algoritmus-animációk hatékony használatát az oktatásban.

Summary

The doctoral dissertation focuses on the design and efficient usage of interactive animations of computer science algorithms in education.

We concentrated, first of all, on the animations of sorting algorithms. However, the collected and summarized suggestions and experiences can be useful for designing, implementing, and effective using of other educationally animations, as well.

The results of pedagogical experiments reported in different publication during the last 30-35 years did not show the effectiveness of using educational animations in every case. In the first part of the doctoral dissertation, we collected those principles and recommendations, which should be used to develop educationally efficient interactive animations. Using theses suggestions we attempt to use virtual worlds for teaching algorithms and programming. Next, we developed a trilingual portal, which contains a collection of algorithm animations from different online sources.

We also created own algorithm animations, mainly interactive animations of sorting algorithm, using Adobe Flash and HTML5/JavaScript technologies. Our animations can be grouped into two broad categories. The first type of animations focuses on the essential aspects, main ideas, and differences of sorting algorithms (e.g. educational games of sorting wooden boxes and playing cards). The second types of animations are micro-level animations, which demonstrate the algorithms in more details (e.g. interactive animations of creating linked lists, basic algorithms on arrays, sorting algorithms). For the easier development of the later animations, we created a JavaScript library („inalan”).

Using our algorithm animations we conducted several pedagogical experiments in the academic years 2014/15 and 2015/16. In these experiments, first-year undergraduate students were involved. We used different statistical methods, using SPSS Statistics, to evaluate the collected data. The statistical tests showed positive results, which prove the effectiveness of the usage of our algorithm animations in education.

Mellékletek

1. melléklet: A diákok tanulási stílusával kapcsolatos kérőív (Google űrlap)
2. melléklet: Virtuális iskolával kapcsolatos kérdőív (Google űrlap)
3. melléklet: Az algoanim.ide.sk portál adatbázisának szerkezete
4. melléklet: Az algoanim.ide.sk portál tárhelyén kialakított mappák szerkezete
5. melléklet: Az interaktív kártyarendezéssel kapcsolatos 1. kérdőív (előtesztelés), 2014/15
6. melléklet: Az interaktív kártyarendezéssel kapcsolatos 2. kérdőív (utótesztelés), 2014/15
7. melléklet: Az interaktív kártyarendezéssel kapcsolatos 1. kérdőív (előtesztelés), 2015/16
8. melléklet: Az interaktív kártyarendezéssel kapcsolatos 2. kérdőív (utótesztelés), 2015/16
9. melléklet: A kártyarendezéssel kapcsolatos 1. tesztre adott válaszok
10. melléklet: A kártyarendezéssel kapcsolatos 2. tesztre adott válaszok
11. melléklet: A ládarendezésnél végrehajtott mérések adatait tároló adatbázis szerkezete
12. melléklet: A ládarendezéssel kapcsolatos kérdőív
13. melléklet: A ládarendezéssel kapcsolatos kérdőívre adott válaszok és a hozzájuk tartozó adatbázisból kinyert adatok
14. melléklet: Különféle buborékrendező algoritmusok és a gyorsrendező algoritmus Java programja
15. melléklet: Dinamikus adatszerkezetekkel kapcsolatos kérdőív (Google űrlap)
16. melléklet: Az „inalan” könyvtár segítségével kialakított animáció forráskódjai
17. melléklet: Előzetes tudás felmérésére szolgáló kérdőív (mindkét csoport)
18. melléklet: Egyszerű cserés rendezéssel kapcsolatos kérdőív (1. csoport - animáció)
19. melléklet: Egyszerű cserés rendezéssel kapcsolatos kérdőív (2. csoport - grafika)

20. melléklet: Buborékrendezéssel kapcsolatos kérdőív (1. csoport - animáció)

21. melléklet: Buborékrendezéssel kapcsolatos kérdőív (2. csoport - grafika)

22. melléklet: Továbbfejlesztett buborékrendezéssel kapcsolatos kérdőív (1. csoport - animáció)

23. melléklet: Továbbfejlesztett buborékrendezéssel kapcsolatos kérdőív (2. csoport - grafika)

24. melléklet: Beszúró rendezéssel kapcsolatos kérdőív (1. csoport - animáció)

25. melléklet: Beszúró rendezéssel kapcsolatos kérdőív (2. csoport - grafika)

26. melléklet: Továbbfejlesztett beszúró rendezéssel kapcsolatos kérdőív (1. csoport - animáció)

27. melléklet: Továbbfejlesztett beszúró rendezéssel kapcsolatos kérdőív (2. csoport - grafika)

28. melléklet: Minimumkiválasztásos rendezéssel kapcsolatos kérdőív (1. csoport - animáció)

29. melléklet: Minimumkiválasztásos rendezéssel kapcsolatos kérdőív (2. csoport - grafika)

30. melléklet: Maximumkiválasztásos rendezéssel kapcsolatos kérdőív (1. csoport - animáció)

31. melléklet: Maximumkiválasztásos rendezéssel kapcsolatos kérdőív (2. csoport - grafika)

32. melléklet: Gyorsrendezéssel kapcsolatos kérdőív (1. csoport - animáció)

33. melléklet: Gyorsrendezéssel kapcsolatos kérdőív (2. csoport - grafika)

34. melléklet: Összefésülő rendezéssel kapcsolatos kérdőív (mindkét csoport)

35. melléklet: Rendezési algoritmusokkal kapcsolatos, 16.-33. mellékletekben található kérdőívekre adott válaszok

1. melléklet: A diákok tanulási stílusával kapcsolatos kérőív (Google űrlap)

Tanulási stílus felmérése

A kérdőív kitöltése mindössze 1-2 percet vesz igénybe, csupán egyetlen lényeges kérdésből áll.

Minden adatot bizalmasan kezelünk, betartva a személyes és adatvédelmi jogokra vonatkozó előírásokat. Személyes adataid harmadik személynek nem adjuk ki semmilyen körülmények között.

Neved anonim azonosítóval helyettesítjük a kérdőív feldolgozása során, jelenlegi válaszod jövőbeli kérdőívekkel való összekapcsolása és a duplicitás kiszűrése miatt szükséges megadni.

Vezetékneved és keresztneved: .....

Évfolyamod:

- ○ 1. évfolyam (Bc.)
- ○ 2. évfolyam (Bc.)
- ○ 3. évfolyam (Bc.)
- ○ 4. évfolyam (Mgr.)
- ○ 5. évfolyam (Mgr.)

Az alábbi kérdéssel azt szeretnénk felmérni, hogy milyen tananyagformát részesítesz előnyben. Nincs helyes vagy helytelen válasz, azt a lehetőséget jelöld be ami Neked a legjobban megfelel.

Ha új algoritmust (pl. tömbelemek rendezése) kellene megtanulnod és az alábbi tananyagformák közül választhatnál, mit részesítenél előnyben?

○ Az algoritmus megértése ANIMÁCIÓ segítségével, pl. weboldalon megjelenített interaktív animáció vagy youtube videó, amely mozgásban mutatja be az algoritmus működését.

○ Az algoritmus megértése STATIKUS KÉPEK alapján, pl. weboldalon vagy könyvben egymás alatt megjelenített képek, melyek szemléltetik az algoritmus lényeges lépéseit.

○ Az algoritmus megértése SZÖVEGES MAGYARÁZAT alapján, pl. weboldalon vagy könyvben olvasható részletes leírás, amely elmagyarázza az algoritmus működését.

2. melléklet: Virtuális iskolával kapcsolatos kérdőív (Google űrlap)

Kérdőív

A kérdőív kitöltésével segíted a Second Life virtuális világban található Luckstone suli fejlesztését, jövőbeli ingyenes tanfolyamok létrejöttét. A kérdőív négy rövid részből áll, kitöltése összesen kb. 4-5 percet vesz igénybe.

Válaszaid anonim módon kerülnek feldolgozásra és publikálásra tudományos kutatás, fejlesztés céljából. Minden adatot bizalmasan kezelünk, betartva a személyes és adatvédelmi jogokra vonatkozó előírásokat. Személyes adataid harmadik személynek nem adjuk ki semmilyen körülmények között.

I. Általános kérdések

1. Second Life felhasználóneved: …….....

Neved anonim azonosítóval helyettesítjük a kérdőív feldolgozása során (jelenlegi válaszaid jövőbeli kérdőívekkel való összekapcsolása és a duplicitás elkerülése miatt szükséges megadni).

2. Nemed (valós életbeli):

○ férfi ○ nő

3. Életkorod (valós életbeli):

○ ... – 19 ○ 20 – 29 ○ 30 – 49 ○ 50 – 69 ○ 70 – ...

II. Second Life használatával kapcsolatos kérdések

1. Milyen tevékenységeket és milyen gyakran szoktál végezni Second Life-ban?

	Soha	Ritkán	Esetenként	Gyakran	Állandóan
Avatárom módosítgatása	○	○	○	○	○
Saját házam elrendezése, módosítása	○	○	○	○	○
Zenehallgatás és táncolás klubokban	○	○	○	○	○

Kiállítások, művészeti galériák meglátogatása	○	○	○	○	○
Új helyek felfedezése	○	○	○	○	○
Vásárlás SL (inworld) üzletekben	○	○	○	○	○
Vásárlás a marketplace-en	○	○	○	○	○
Szöveges (text) csevegés	○	○	○	○	○
Csevegés mikrofonon keresztül (voice)	○	○	○	○	○
Szerepjátékokban való részvétel	○	○	○	○	○
Saját 3D objektumok létrehozása	○	○	○	○	○
Saját LSL szkriptek írása	○	○	○	○	○
Részvétel építéssel kapcsolatos tanfolyamokon	○	○	○	○	○
Idegen nyelv tanulása, gyakorlása	○	○	○	○	○

III. Modellezéssel, programozással kapcsolatos kérdések

1. Mennyire tudsz alapobjektumokból építkezni és azokat textúrázni?

Egyáltalán nem tudok 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ Bármit elkészítek

2. Mennyire tudsz 3D modelleket készíteni külső modellező szoftverek segítségével?

Egyáltalán nem tudok 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ Bármit elkészítek

3. Amennyiben már használtál 3D modellező szoftvert, milyeneket?

(pl. Blender, Maya, 3D Studio Max, …) …….

4. Mennyire tudsz programozni (tetszőleges programnyelven)?

Egyáltalán nem tudok 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ Bármit elkészítek

5. Amennyiben már programoztál, milyen programnyelven?

(pl. Pascal, C, C++, C#, Java, Python, PHP, ...) …….

6. Mennyire tudsz LSL szkripteket írni?

Egyáltalán nem tudok 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ Bármit elkészítek

IV. Jövőbeli tanfolyamokkal kapcsolatos kérések

1. Szeretnél részt venni magyar nyelven folyó, ingyenes online tanfolyamokon, óránkon Second Life-ban?

○ Igen ○ Nem

2. Milyen témájú tanfolyamok érdekelnének?

(pl. 3D modellezés, Blender, Programzás, Pascal, LSL szkriptek, Angol, Könyvklub, ...)

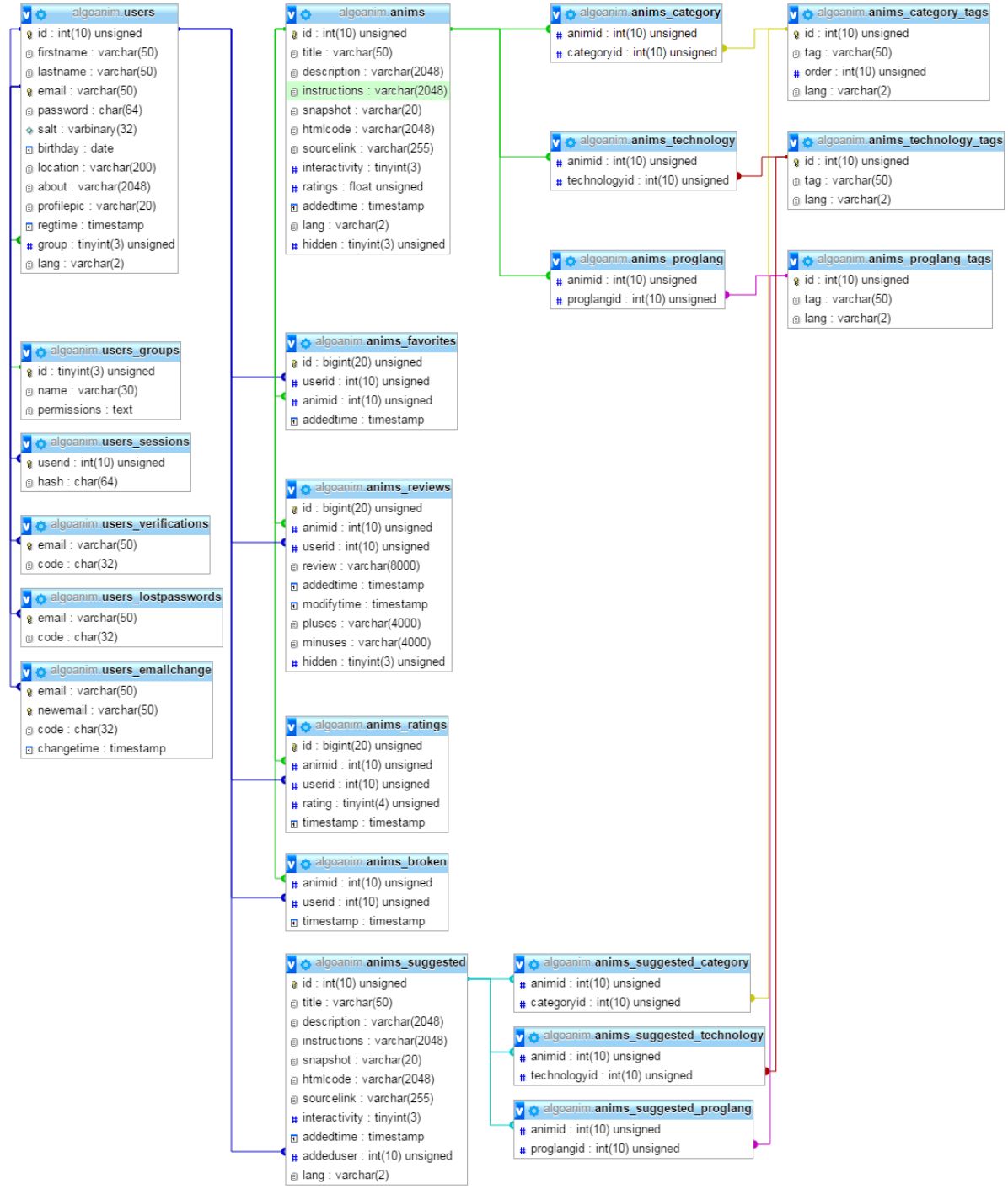
……

3. Megjegyzésed a kérdőívvel kapcsolatban:

……

Válaszaid rögzítettük. Köszönjük a kérdőív kitöltését és ezzel a kutatás és virtuális világbeli oktatás fejlesztését.

3. melléklet: Az algoanim.ide.sk portál adatbázisának szerkezete



4. melléklet: Az algoanim.ide.sk portál tárhelyén kialakított mappák szerkezete

- **classes** – osztályok PHP államányait tartalmazza (Anim, AnimList, AnimSuggested, AnimSuggestedList, Categories, Config, Cookie, DB, Hash, Input, Language, Mail, Redirect, Reviews, ReviewsView, Session, Token, User, Validator, View),
    - **phpmailer –** PHPMailer könyvtár állományai e-mailek küldéséhez,
- **core –** a portál beállításait tartalmazó konfigurációs állományok,
- **css –** stíluslapokat tartalmazó mappa,
    - **css_anims –** animációkkal kapcsolatos weboldalakhoz tartozó stíluslapok,
    - **css_users –** felhasználókkal kapcsolatos weboldalakhoz tartozó stíluslapok,
- **functions –** általános JavaScript függvények mappája,
- **imgs –** képeket tartalmazó mappa,
    - **anims –** animációk képernyőképeit tartalmazó mappa,
    - **links –** más weboldalak képernyőképeit tartalmazó mappa,
    - **profilepics –** felhasználói profilképeket tartalmazó mappa,
- **includes –** weboldalrészeket tartalmazó mappa,
    - **inc_anims –** animációkkal kapcsolatos weboldalakhoz tartozó részek,
    - **inc_users –** felhasználókkal kapcsolatos weboldalakhoz tartozó részek,
        - **useraccount –** felhasználói fiókokkal kapcsolatos weboldalakhoz tartozó részek (profil, e-mail cím, jelszómódosítás),
    - **mails –** a portál által küldött automatikus HTML és TXT formátumú e-mailek szövegeit tartalmazó mappa,
- **js** – animációk megjelenítéséhez és felhasználói vélemények írásához használt JavaScript állományok,
- **languages** – a portál szöveges részeit különböző nyelveken,
    - **en –** angol nyelvi fájlok,
    - **hu –** magyar nyelvi fájlok,
    - **sk –** szlovák nyelvi fájlok.

5. melléklet: Az interaktív kártyarendezéssel kapcsolatos 1. kérdőív (előtesztelés), 2014/15

Próbáld meg megoldani az alábbi feladatokat az eddigi ismereteid alapján!

Az alábbi táblázatban felsorolt rendezési algoritmusok az elemeket **NÖVEKVŐ SORRENDBE** rendezik, a legkisebb elemtől a legnagyobbig.

Mindegyik rendezési algoritmus oszlopában jelöld be **X**-el az algoritmusra jellemző kijelentéseket. Az egyes rendezési algoritmusokra több kijelentés is lehet jellemző.

	Egyszerű cserés rendezés (Simplesort)	**Buborékrendezés** (Bubblesort)	**Beszúró rendezés** (Insertion sort)	**Minimumkiválasztásos rendezés** (Selection sort: Minsort)	**Maximumkiválasztásos rendezés** (Selection sort: Maxsort)
Mindig egymás melletti két szomszédos elemet hasonlítunk össze.					
Mindegyik elemet összehasonlítjuk az összes mögötte levővel.					
Mindig előbb végignézzük a rendezetlen elemeket, melyek közül kiválasztunk egyet, majd ezt az elemet cseréljük fel a tömbrész elején vagy végén levő elemmel.					
Minden egyes végigfutásnál a **legkisebb elem** a rendezetlen rész **elejére** kerül (a rendezett sorrend a tömb elejétől kezd kialakulni).					
Minden egyes végigfutásnál a **legnagyobb elem** a rendezetlen rész **végére** kerül (a rendezett sorrend a tömb végétől kezd kialakulni).					
A rendezés alatt, a tömb elején vagy végén kialakult rendezett részben az elemek **már nem módosulnak (nem tolódnak arrébb)** a rendezés befejezéséig.					
A rendezés alatt, a tömb elején vagy végén kialakult rendezett részben az elemek **még módosulhatnak (arrébb tolódhatnak)** a rendezés befejezéséig.					

A pszeudokódokhoz rendeld hozzá az alábbi rendezési algoritmusokat.
Mindegyik algoritmust csak egy pszeudokódhoz rendelhetsz hozzá!

A - Egyszerű cserés rendezés (Simplesort)

B - Buborékrendezés (Bubblesort)

C - Beszúró rendezés (Insertion sort)

D - Minimumkiválasztásos rendezés (Selection sort: Minsort)

E - Maximumkiválasztásos rendezés (Selection sort: Maxsort)

```
ciklus I = N-től 2-ig
   MAX = I
   ciklus J = 1-től I-ig
      ha A[J]>A[MAX] akkor
            MAX = J
   csere A[I],A[MAX]
```

.....

```
ciklus I = N-1-től 1-ig
   ciklus J = 1-től I-ig
      ha A[J]>A[J+1] akkor
            csere A[J],A[J+1]
```

.....

```
ciklus I = 1-től N-1-ig
   ciklus J = I+1-től N-ig
      ha A[I]>A[J] akkor
            csere A[I],A[J]
```

.....

```
ciklus I = 1-től N-1-ig
   MIN = I
   ciklus J = I+1-től N-ig
      ha A[J]<A[MIN] akkor
            MIN = J
   csere A[I],A[MIN]
```

.....

```
ciklus I = 2-től N-ig
   J = I - 1
   ciklus amíg J>0 és A[J]>A[J+1]
      csere A[J],A[J+1]
      J = J - 1
```

.....

6. melléklet: Az interaktív kártyarendezéssel kapcsolatos 2. kérdőív (utótesztelés), 2014/15

Látogass el a **http://rendezes.ide.sk** weboldalra, majd az ott található kártyalapok rendezésével próbáld meg újra átgondolni és megoldani az alábbi feladatokat!

Az alábbi táblázatban felsorolt rendezési algoritmusok az elemeket **NÖVEKVŐ SORRENDBE** rendezik, a legkisebb elemtől a legnagyobbig.

Mindegyik rendezési algoritmus oszlopában jelöld be **X**-el az algoritmusra jellemző kijelentéseket. Az egyes rendezési algoritmusokra több kijelentés is lehet jellemző.

	Egyszerű cserés rendezés (Simplesort)	**Buborékrendezés** (Bubblesort)	**Beszúró rendezés** (Insertion sort)	**Minimumkiválasztásos rendezés** (Selection sort: Minsort)	**Maximumkiválasztásos rendezés** (Selection sort: Maxsort)
Mindig egymás melletti két szomszédos elemet hasonlítunk össze.					
Mindegyik elemet összehasonlítjuk az összes mögötte levővel.					
Mindig előbb végignézzük a rendezetlen elemeket, melyek közül kiválasztunk egyet, majd ezt az elemet cseréljük fel a tömbrész elején vagy végén levő elemmel.					
Minden egyes végigfutásnál a **legkisebb elem** a rendezetlen rész **elejére** kerül (a rendezett sorrend a tömb elejétől kezd kialakulni).					
Minden egyes végigfutásnál a **legnagyobb elem** a rendezetlen rész **végére** kerül (a rendezett sorrend a tömb végétől kezd kialakulni).					
A rendezés alatt, a tömb elején vagy végén kialakult rendezett részben az elemek **már nem módosulnak (nem tolódnak arrébb)** a rendezés befejezéséig.					
A rendezés alatt, a tömb elején vagy végén kialakult rendezett részben az elemek **még módosulhatnak (arrébb tolódhatnak)** a rendezés befejezéséig.					

A pszeudokódokhoz rendeld hozzá az alábbi rendezési algoritmusokat.
Mindegyik algoritmust csak egy pszeudokódhoz rendelhetsz hozzá!

A - Egyszerű cserés rendezés (Simplesort)

B - Buborékrendezés (Bubblesort)

C - Beszúró rendezés (Insertion sort)

D - Minimumkiválasztásos rendezés (Selection sort: Minsort)

E - Maximumkiválasztásos rendezés (Selection sort: Maxsort)

```
ciklus I = N-től 2-ig
   MAX = I
   ciklus J = 1-től I-ig
      ha A[J]>A[MAX] akkor
            MAX = J
   csere A[I],A[MAX]
```

.....

```
ciklus I = N-1-től 1-ig
   ciklus J = 1-től I-ig
      ha A[J]>A[J+1] akkor
            csere A[J],A[J+1]
```

.....

```
ciklus I = 1-től N-1-ig
   ciklus J = I+1-től N-ig
      ha A[I]>A[J] akkor
            csere A[I],A[J]
```

.....

```
ciklus I = 1-től N-1-ig
   MIN = I
   ciklus J = I+1-től N-ig
      ha A[J]<A[MIN] akkor
            MIN = J
   csere A[I],A[MIN]
```

.....

```
ciklus I = 2-től N-ig
   J = I - 1
   ciklus amíg J>0 és A[J]>A[J+1]
      csere A[J],A[J+1]
      J = J - 1
```

.....

Most értékeld a **http://rendezes.ide.sk** oldalon található interaktív animációkat.

Ez a papírlap szét lesz választva az előzőtől és az itt megadott válaszok csak ezután, névtelenül lesznek kiértékelve. Ezért kérjük, hogy erre a papírlapra ne írj nevet és a kérdésekre őszintén válaszolj!

1. A weboldalon található animációk érthetőek, világosak?

Egyszerű cserés rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Buborékrendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Beszúró rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Minimumkiválasztásos rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Maximumkiválasztásos rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

2. A kezelésük az egér segítségével könnyen elsajátítható?

Egyszerű cserés rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Buborékrendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Beszúró rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Minimumkiválasztásos rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Maximumkiválasztásos rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

3. A szemléletességük grafikailag megfelelő szinten van?

Egyszerű cserés rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Buborékrendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Beszúró rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Minimumkiválasztásos rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

Maximumkiválasztásos rendezés

egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, teljes mértékben*

4. Egyéb észrevételek

Volt bármi zavaró az animációkban, megértésükben, kezelésükben, stb.?

Ha volt ilyen, kérjük, írd le ide saját szavaiddal mi volt az.

Amennyiben csak némelyik rendezési algoritmusnál volt ilyen,
ne felejtsd el odaírni azt is, hogy melyik algoritmusnál.

7. melléklet: Az interaktív kártyarendezéssel kapcsolatos 1. kérdőív (előtesztelés), 2015/16

Próbáld meg megoldani az alábbi feladatokat az eddigi ismereteid alapján!

Az alábbi táblázatban felsorolt rendezési algoritmusok az elemeket **NÖVEKVŐ SORRENDBE** rendezik, a legkisebb elemtől a legnagyobbig.

Mindegyik rendezési algoritmus oszlopában jelöld be **X**-el az algoritmusra jellemző kijelentéseket. Az egyes rendezési algoritmusokra több kijelentés is lehet jellemző.

	Egyszerű cserés rendezés (Simplesort)	**Buborékrendezés** (Bubblesort)	**Beszúró rendezés** (Insertion sort)	**Minimumkiválasztásos rendezés** (Selection sort: Minsort)	**Maximumkiválasztásos rendezés** (Selection sort: Maxsort)
Mindig egymás melletti két szomszédos elemet hasonlítunk össze.					
Mindegyik elemet összehasonlítjuk az összes mögötte levővel.					
Mindig előbb végignézzük a rendezetlen elemeket, melyek közül kiválasztunk egyet, majd ezt az elemet cseréljük fel a tömbrész elején vagy végén levő elemmel.					
Minden egyes végigfutásnál a **legkisebb elem** a rendezetlen rész **elejére** kerül.					
Minden egyes végigfutásnál a **legnagyobb elem** a rendezetlen rész **végére** kerül.					
A rendezés alatt, a tömb elején vagy végén kialakult rendezett részben az elemek **már nem módosulnak (nem tolódnak arrébb)** a rendezés befejezéséig.					
A rendezés alatt, a tömb elején vagy végén kialakult rendezett részben az elemek **még módosulhatnak (arrébb tolódhatnak)** a rendezés befejezéséig.					

A pszeudokódokhoz rendeld hozzá az alábbi rendezési algoritmusokat.
Mindegyik algoritmust csak egy pszeudokódhoz rendelhetsz hozzá!

A - Egyszerű cserés rendezés (Simplesort)

B - Buborékrendezés (Bubblesort)

C - Beszúró rendezés (Insertion sort)

D - Minimumkiválasztásos rendezés (Selection sort: Minsort)

E - Maximumkiválasztásos rendezés (Selection sort: Maxsort)

```
ciklus I = N-től 2-ig
   M = I
   ciklus J = 1-től I-ig
      ha A[J]>A[M] akkor
            M = J
   csere A[I],A[M]
```

.....

```
ciklus I = N-1-től 1-ig
   ciklus J = 1-től I-ig
      ha A[J]>A[J+1] akkor
            csere A[J],A[J+1]
```

.....

```
ciklus I = 1-től N-1-ig
   ciklus J = I+1-től N-ig
      ha A[I]>A[J] akkor
            csere A[I],A[J]
```

.....

```
ciklus I = 1-től N-1-ig
   M = I
   ciklus J = I+1-től N-ig
      ha A[J]<A[M] akkor
            M = J
   csere A[I],A[M]
```

.....

```
ciklus I = 2-től N-ig
   J = I - 1
   ciklus amíg J>0 és A[J]>A[J+1]
      csere A[J],A[J+1]
      J = J - 1
```

.....

8. melléklet: Az interaktív kártyarendezéssel kapcsolatos 2. kérdőív (utótesztelés), 2015/16

Látogass el a **http://rendezes.ide.sk** weboldalra, majd az ott található kártyalapok rendezésével próbáld meg újra átgondolni és megoldani az alábbi feladatokat!

Az alábbi táblázatban felsorolt rendezési algoritmusok az elemeket **NÖVEKVŐ SORRENDBE** rendezik, a legkisebb elemtől a legnagyobbig.

Mindegyik rendezési algoritmus oszlopában jelöld be **X**-el az algoritmusra jellemző kijelentéseket. Az egyes rendezési algoritmusokra több kijelentés is lehet jellemző.

	Egyszerű cserés rendezés (Simplesort)	**Buborékrendezés** (Bubblesort)	**Beszúró rendezés** (Insertion sort)	**Minimumkiválasztásos rendezés** (Selection sort: Minsort)	**Maximumkiválasztásos rendezés** (Selection sort: Maxsort)
Mindig egymás melletti két szomszédos elemet hasonlítunk össze.					
Mindegyik elemet összehasonlítjuk az összes mögötte levővel.					
Mindig előbb végignézzük a rendezetlen elemeket, melyek közül kiválasztunk egyet, majd ezt az elemet cseréljük fel a tömbrész elején vagy végén levő elemmel.					
Minden egyes végigfutásnál a **legkisebb elem** a rendezetlen rész **elejére** kerül.					
Minden egyes végigfutásnál a **legnagyobb elem** a rendezetlen rész **végére** kerül.					
A rendezés alatt, a tömb elején vagy végén kialakult rendezett részben az elemek **már nem módosulnak (nem tolódnak arrébb)** a rendezés befejezéséig.					
A rendezés alatt, a tömb elején vagy végén kialakult rendezett részben az elemek **még módosulhatnak (arrébb tolódhatnak)** a rendezés befejezéséig.					

A pszeudokódokhoz rendeld hozzá az alábbi rendezési algoritmusokat.
Mindegyik algoritmust csak egy pszeudokódhoz rendelhetsz hozzá!

A - Egyszerű cserés rendezés (Simplesort)

B - Buborékrendezés (Bubblesort)

C - Beszúró rendezés (Insertion sort)

D - Minimumkiválasztásos rendezés (Selection sort: Minsort)

E - Maximumkiválasztásos rendezés (Selection sort: Maxsort)

```
ciklus I = N-től 2-ig
   M = I
   ciklus J = 1-től I-ig
      ha A[J]>A[M] akkor
            M = J
   csere A[I],A[M]
```

.....

```
ciklus I = N-1-től 1-ig
   ciklus J = 1-től I-ig
      ha A[J]>A[J+1] akkor
            csere A[J],A[J+1]
```

.....

```
ciklus I = 1-től N-1-ig
   ciklus J = I+1-től N-ig
      ha A[I]>A[J] akkor
            csere A[I],A[J]
```

.....

```
ciklus I = 1-től N-1-ig
   M = I
   ciklus J = I+1-től N-ig
      ha A[J]<A[M] akkor
            M = J
   csere A[I],A[M]
```

.....

```
ciklus I = 2-től N-ig
   J = I - 1
   ciklus amíg J>0 és A[J]>A[J+1]
      csere A[J],A[J+1]
      J = J - 1
```

.....

9. melléklet: A kártyarendezéssel kapcsolatos 1. tesztre adott válaszok

diák sorszáma	A (simplesort) állítások:							B (bubblesort) állítások:							C (insertsort) állítások:							D (minsort) állítások:							E (maxsort) állítások:							pszeudokódok párosítása				
	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	maxsort	simplesort	insertsort	bubblesort	minsort
1.		x													x								x							x							e	a	b	c	d
2.						x		x		x			x								x		x		x								x				e	c	b	a	d
3.		x				x				x				x	x										x								x				e	a	b	c	d
4.	x	x					x	x						x			x								x		x						x	x		e	a	c	b	d
5.	x			x					x	x				x							x	x	x		x		x		x	x			x	x		e	a	b	c	d
6.	x			x		x		x				x	x				x				x		x		x		x			x			x	x		e	a	c	b	d
7.		x						x						x											x		x						x	x		e	a	c	b	d
8.		x				x		x	x				x				x				x				x		x				x		x	x		e	b	c	a	d
9.								x	x				x				x				x				x								x			e	a	c	b	d
10.	x					x		x									x				x		x		x					x			x			e	a	b	c	d
11.																																				e	b	a	c	d
12.		x				x		x	x					x										x	x		x				x		x	x		e	b	c	a	d
13.	x		x	x					x		x		x								x			x	x			x			x		x		x	e	b	c	a	d
14.		x					x	x						x			x				x			x	x		x				x		x	x		e	a	c	b	d
15.	x												x					x			x		x								x		x			b	a	d	e	c
16.	x							x	x					x			x								x		x						x	x		e	b	c	a	d
17.	x					x		x									x				x		x		x					x			x			e	b	c	a	d
18.		x					x	x						x	x					x				x	x						x			x			e	b	c	a	d
19.	x	x						x	x					x			x								x		x	x					x	x	x	e	a			d
20.							x	x						x	x						x				x	x						x		x			e	a	c	b	d
21.	x							x		x				x	x						x			x	x	x						x		x	x	x	e	a	b	c	d
22.	x						x		x								x				x				x		x						x	x		e	b	c	a	d
23.		x					x	x						x			x							x	x						x		x			e	a	c	b	d
24.		x										x			x										x								x			e	c	b	a	d
25.		x				x		x						x			x				x					x						x				e	c	a	b	d
26.		x						x									x								x								x			e	a	c	b	d
27.		x					x							x	x						x			x	x		x				x		x	x		e	a	c	b	d
28.		x		x		x		x				x					x				x				x								x			e	a	c	b	d
29.								x																	x								x			e	a	c	b	d
30.	x					x							x				x				x		x		x		x			x			x	x		e	b	a	c	d
31.	x								x					x			x								x		x						x	x		e	b	c	a	d
32.	x	x					x	x									x								x		x						x	x		e	b	c	a	d
33.	x	x			x	x	x			x	x	x									x				x		x						x	x		e	c	b	a	d
34.		x					x	x		x				x							x				x		x						x	x		e	c	a	b	d
35.		x					x	x						x			x				x				x		x						x	x		e	c	a	b	d
36.		x				x	x	x									x				x				x								x			e	b	c	a	d
37.							x			x												x	x		x		x		x	x			x	x		e	b	a	c	d
38.		x						x						x		x				x	x			x	x						x			x			e	a	c	b	d
39.		x				x		x			x	x					x				x				x								x			e	a	c	b	d
40.	x			x		x		x			x			x			x		x	x			x			x	x			x			x	x		d	a	c	b	e
41.		x		x		x		x				x	x		x			x			x			x	x		x				x		x	x		e	a	c	b	d
42.	x						x	x	x		x			x			x				x		x	x	x		x			x	x		x	x		b	a	e	d	c
43.	x							x								x								x	x		x											b		a	
44.	x					x		x					x			x					x			x	x								x				e			a	d
45.		x						x					x											x	x		x				x		x	x		e	a	c	b	d
46.			x	x			x	x						x		x	x	x	x	x				x	x	x		x			x	x		x	x		d	a	b	c	e
47.	x			x		x		x					x										x	x	x			x			x	x		x		x	e	b	c	a	d
48.	x			x		x		x					x										x	x	x			x			x	x		x		x	e	b	c	a	d
49.	x	x		x		x		x				x	x											x	x			x				x		x			d	a	c	b	e

50.	x							x								x	x				x			x	x		x				x		x	x		d	c	b	a	e
51.	x	x					x	x	x					x			x			x					x		x						x	x		d				e
52.		x		x		x		x				x	x		x			x		x			x	x	x			x		x	x		x		x	d	a	c	b	e
53.		x				x		x					x			x					x		x	x	x		x			x	x		x	x		e	a	c	b	d
54.		x		x		x		x				x	x		x			x			x			x	x		x				x		x	x		e	a	c	b	d
55.		x	x				x	x						x	x	x	x			x						x	x				x	x		x		d	a	b	c	e
56.	x			x		x		x											x	x	x	x			x		x	x	x		x		x			b	d	e	a	c
57.	x			x		x		x				x	x											x	x			x		x	x		x		x	b	d	e	a	c
58.	x			x			x	x			x	x	x			x					x		x	x	x		x				x		x	x		e	a	c	b	d
59.	x						x	x			x	x	x			x		x			x		x	x	x		x				x		x	x		e	a	c	b	d
60.	x					x		x	x				x								x			x	x			x			x		x		x	c	a	d	b	e
61.	x							x									x				x			x	x		x				x		x	x		a	d	e	b	c
62.		x		x		x		x			x		x			x	x	x		x				x	x		x				x		x	x		d	a	c	b	e
63.		x		x				x													x			x	x		x				x		x			e	a	c	b	d
64.	x						x	x			x		x	x		x		x	x				x	x	x		x				x		x	x		e	a	c	b	d
65.		x						x					x								x				x						x		x			d	a	c	b	e
66.	x			x		x		x			x		x			x			x		x		x	x	x		x				x		x	x		e	a	d	b	c
67.	x	x				x		x	x				x				x				x					x						x				e	a	c	b	d
68.		x						x					x								x				x						x		x			d	a	c	b	e
69.	x			x								x		x		x				x				x	x						x		x			d	a	c	b	e
70.	x			x		x						x		x									x	x	x					x	x		x		x	d	a	c	b	e
71.	x			x		x		x				x	x				x				x			x	x		x						x	x		e	a	c	b	d
72.	x			x		x		x			x		x			x	x	x			x		x	x		x		x		x	x		x		x	e	a	c	b	d
73.		x		x		x		x				x		x	x			x			x			x	x		x				x		x	x		d	a	c	b	e
74.	x	x					x	x			x		x		x					x						x										d	a	c	b	e
75.	x	x		x		x		x						x	x		x			x				x	x			x			x				x	d	a	c	b	e
76.				x			x	x			x					x			x					x			x				x		x	x		c	b	e	a	d
77.	x			x					x					x		x			x	x		x		x							x		x	x		a	c		b	
78.		x		x		x		x				x		x		x	x	x			x		x			x	x			x			x	x		d	a	c	b	e
79.	x						x					x	x										x	x	x		x			x	x			x		e	a	c	b	d
80.	x			x				x			x		x		x			x			x	x		x	x				x	x	x		x			d	c	e	a	b
81.								x																x										x		e	a	c	b	d
82.								x																												e	a	c	b	d
83.	x			x		x		x							x						x			x						x	x					e	a	c	b	d
84.	x			x		x		x							x					x	x			x	x		x			x	x		x	x		e	a	c	b	d
85.				x		x		x				x	x		x						x			x	x		x			x	x		x	x		e	a	c	b	d
86.	x	x					x	x						x						x				x	x								x			b	a	c	e	d
87.		x		x			x	x			x			x	x			x			x			x	x		x				x		x	x		e	a	c	b	d
88.	x					x							x		x		x						x	x	x			x		x	x		x		x	e	a	c	b	d
89.	x			x		x						x	x		x			x		x			x	x	x			x		x	x		x		x	e	b	c	a	d
90.		x		x			x	x				x		x		x			x	x			x	x	x		x			x	x		x	x		d	a	c	b	e
91.	x					x		x					x				x	x		x			x	x	x			x		x	x		x		x	a	d	c	e	b
92.		x		x		x		x	x			x	x		x					x				x				x			x				x	d	a	c	b	e

Helyes válaszok:

–		x		x		x		x				x	x		?			?			x			x	x		x				x		x	x		e	a	c	b	d

Diákok sorszáma:

- 2014/15-ös évben végzett tesztelések: 1. – 39.
- 2015/16-os évben végzett tesztelések: 40. – 92.

10. melléklet: A kártyarendezéssel kapcsolatos 2. tesztre adott válaszok

diák sorszáma	A (simplesort) állítások:							B (bubblesort) állítások:							C (insertsort) állítások:							D (minsort) állítások:							E (maxsort) állítások:							pszeudokódok párosítása				
	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	maxsort	simplesort	insertsort	bubblesort	minsort
1.		x		x		x		x				x	x		x			x			x			x	x		x			x		x	x		e	a	c	b	d
2.		x		x		x		x				x	x			x					x			x	x		x			x		x	x		e	c	b	a	d
3.	x			x			x	x	x			x		x		x					x			x	x		x			x		x	x		e	a	c	b	d
4.				x		x		x					x					x			x			x	x		x			x		x	x		e	c	a	b	d
5.		x		x		x		x				x	x								x			x	x		x			x		x	x		e	a	c	b	d
6.			x	x		x		x				x	x			x		x			x			x	x		x			x		x	x		e	a	c	b	d
7.		x		x		x		x				x		x				x	x					x	x		x			x		x	x		e	a	c	b	d
8.		x		x		x		x				x	x			x					x			x	x		x			x		x	x		e	b	c	a	d
9.				x			x					x		x		x		x			x		x	x	x		x			x		x	x		e	a	b	c	d
10.		x				x		x				x	x		x									x	x		x			x		x	x		e	a	c	b	d
11.								x									x				x				x								x		e	a	b	c	d
12.		x		x		x		x				x	x				x				x			x	x		x			x		x	x		e	b	c	a	d
13.		x		x		x		x				x	x		x			x			x		x	x	x		x		x	x		x	x		e	a	c	b	d
14.		x	x	x		x		x				x	x					x			x			x	x		x			x		x	x		e	a	c	b	d
15.		x		x		x		x				x	x		x			x			x		x	x	x		x								e		b		d
16.			x					x				x	x			x					x			x	x		x		x	x		x	x		e	b	c	a	d
17.		x				x		x					x								x			x	x		x			x		x	x		e	a	c	b	d
18.		x		x		x		x				x	x			x					x			x	x		x			x		x	x		e	b	c	a	d
19.				x				x	x						x		x				x					x	x					x	x						
20.		x		x		x		x				x	x		x						x			x	x		x			x		x	x		e	a	c	b	d
21.			x	x		x		x				x	x			x					x			x	x		x			x		x	x		e	a	c	b	d
22.		x		x		x		x	x			x	x					x			x			x	x		x			x		x	x		e	a	b	c	d
23.		x		x			x	x				x	x				x	x			x			x	x		x			x		x	x		e	a	c	b	d
24.		x		x		x		x				x	x		x			x			x		x	x	x		x		x	x		x	x		e	a	b	c	d
25.		x		x		x		x				x	x			x	x		x		x			x	x		x			x		x	x		e	c	a	b	d
26.		x		x		x		x					x								x			x	x		x			x		x	x		e	a	c	b	d
27.		x		x		x		x		x				x							x				x		x						x	x		e	c	b	a	d
28.		x		x		x		x				x	x					x			x			x	x		x			x		x	x		e	a	c	b	d
29.			x	x		x		x				x	x			x					x				x		x						x	x		e	a	c	b	d
30.		x		x		x		x				x	x				x				x				x		x						x	x		e	a	b	c	d
31.		x		x		x		x				x	x					x			x			x	x		x			x		x	x		e	a	b	c	d
32.		x				x		x					x			x	x				x				x		x						x	x		e	b	c	a	d
33.		x		x		x		x				x	x				x				x			x	x		x			x		x	x		e	a	b	c	d
34.		x		x		x		x				x	x				x				x			x	x	x		x		x	x		x		x	e	a	b	c	d
35.		x		x		x		x				x	x				x				x			x	x		x			x		x	x		e	a	b	c	d
36.		x		x		x		x				x	x		x						x			x	x		x			x		x	x		e	a	b	c	d
37.		x		x		x		x				x	x				x				x			x	x		x			x		x	x		e	a	b	c	d
38.			x	x				x					x			x					x				x		x						x	x		e	b	c	a	d
39.		x	x					x			x	x	x				x				x				x								x			e	a	c	b	d
40.		x		x		x		x				x	x				x	x			x				x		x				x		x	x		d	a	c	b	e
41.		x		x		x		x				x	x		x						x				x		x				x		x	x		e	a	c	b	d
42.		x		x		x		x				x	x			x		x			x				x		x				x		x	x		d	a	c	b	e
43.		x	x	x		x		x			x			x				x			x				x		x						x	x			b		a	
44.		x	x	x			x	x				x	x					x			x												x					b	
45.		x		x		x		x				x	x								x				x		x				x		x	x		e	a	c	b	d
46.		x		x		x		x			x						x		x		x				x						x		x			d	a	b	c	e
47.		x					x	x				x	x		x			x			x		x	x	x		x						x	x		e	a	c	b	d
48.		x					x	x				x	x		x			x			x		x		x		x			x	x		x	x		d	a	b	c	e

49.		x		x		x		x				x	x				x	x			x		x	x	x			x			x				x	e	a	c	b	d
50.		x		x		x		x			x			x			x	x			x		x	x	x			x		x	x		x		x	e	a	c	b	d
51.		x		x			x	x				x					x	x			x			x	x		x				x		x	x		d				e
52.		x		x		x		x				x	x				x				x			x	x			x			x		x		x	e	a	c	b	d
53.		x		x		x		x				x	x								x			x	x		x				x		x	x		e	a	c	b	d
54.		x		x		x		x				x		x			x	x			x		x		x		x			x			x	x		e	a	c	b	d
55.		x		x			x	x				x	x			x	x				x		x	x	x		x			x	x			x		d	c	b	a	e
56.			x	x			x	x			x			x		x			x		x			x	x		x				x		x	x		b	d	e	a	c
57.			x	x			x	x			x			x	x	x			x		x			x	x		x				x		x	x		b	d	e	a	c
58.		x		x		x		x				x	x			x		x			x			x	x		x				x		x	x		e	a	c	b	d
59.		x		x		x		x				x	x		x	x		x			x		x	x	x		x				x		x	x		e	a	c	b	d
60.		x		x		x		x				x	x				x				x		x		x		x			x			x	x		d	a	b	c	e
61.				x		x		x				x	x			x									x		x						x	x		b	a	c	e	d
62.		x	x	x		x		x				x	x			x		x	x		x			x	x		x				x		x	x		d	a	c	b	e
63.		x		x		x		x				x	x								x			x	x		x				x		x	x		d	a	c	b	e
64.		x		x		x		x				x	x								x			x	x		x				x		x	x		d	a	c	b	e
65.		x				x		x				x					x				x				x		x						x	x		d	a	c	b	e
66.		x		x		x		x				x	x			x		x			x			x	x		x				x		x	x		e	a	c	b	d
67.						x		x			x			x		x					x		x			x		x		x		x			x	e	a	c	b	d
68.		x				x		x				x					x				x				x		x						x	x		d	a	c	b	e
69.		x	x	x		x		x				x	x					x			x			x	x		x				x		x	x		d	a	c	b	e
70.		x	x	x		x		x				x	x			x		x	x		x			x	x	x	x						x	x		d	a	c	b	e
71.		x		x		x		x				x	x				x	x			x			x	x		x				x		x	x		d	a	c	b	e
72.		x		x		x		x				x	x				x		x		x			x	x		x				x		x	x		d	a	c	b	e
73.		x		x		x		x				x		x	x			x			x			x	x		x				x		x	x		d	a	c	b	e
74.		x		x		x		x				x		x	x						x			x			x				x			x		d	a	c	b	e
75.		x		x		x		x		x		x	x			x	x				x			x			x							x		d	a	c	b	e
76.		x		x			x	x				x		x		x			x	x				x	x		x				x		x	x		e	a	c	b	d
77.		x		x		x		x					x				x				x		x		x								x		x	a	b	e	d	c
78.		x		x			x	x			x			x		x	x	x			x		x	x	x		x			x	x		x	x		e	a	c	b	d
79.		x		x		x		x				x	x			x		x			x			x	x		x				x		x	x		e	a	c	b	d
80.			x	x		x		x				x	x			x		x			x		x	x	x		x			x	x		x	x		e	b	c	a	d
81.		x			x	x		x					x					x			x			x			x				x			x		e	b	c	a	d
82.		x			x	x		x			x		x						x		x			x			x				x			x		e	b	c	a	d
83.	x	x			x	x		x			x		x						x		x			x						x	x					e	b	c	a	d
84.					x		x	x			x			x		x			x					x			x				x			x		e	b	c	a	d
85.					x	x		x								x					x		x	x	x		x			x	x		x	x		e	a	c	b	d
86.						x		x							x			x		x				x	x		x				x			x		d	a	b	c	e
87.		x	x	x		x		x				x	x		x			x			x		x	x	x		x			x	x		x	x		e	a	c	b	d
88.		x		x		x		x				x	x				x	x			x			x	x		x				x		x	x		a	c	d	e	b
89.			x	x		x		x				x	x		x			x		x			x		x		x			x			x	x		d	a	c	b	e
90.		x		x		x		x				x	x			x	x	x			x		x	x	x			x		x	x		x		x	e	a	c	b	d
91.				x		x		x				x	x								x		x	x	x		x			x	x		x	x		a	e	b	d	c
92.				x		x		x				x	x			x		x			x			x	x		x				x		x	x		d	a	c	b	e

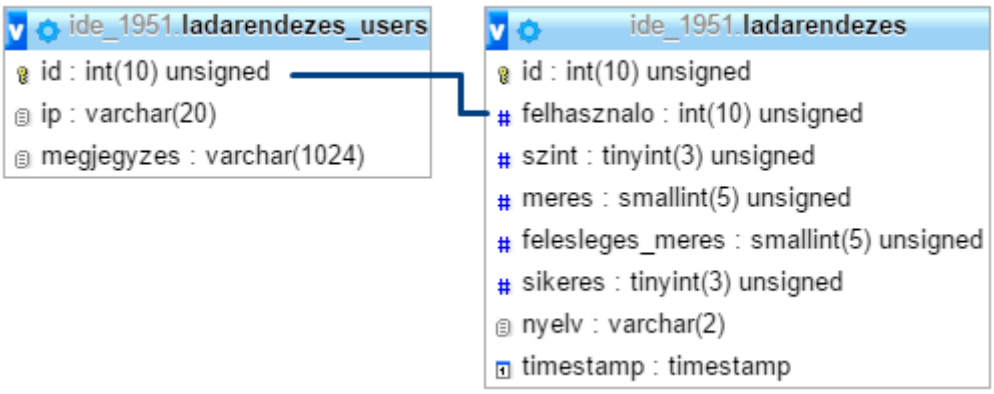
Helyes válaszok:

–		x		x		x		x				x	x		?			?			x			x	x		x				x		x	x		e	a	c	b	d

Diákok sorszáma:

- 2014/15-ös évben végzett tesztelések: 1. – 39.
- 2015/16-os évben végzett tesztelések: 40. – 92.

11. melléklet: A ládarendezésnél végrehajtott mérések adatait tároló adatbázis szerkezete



12. melléklet: A ládarendezéssel kapcsolatos kérdőív

1. Kérjük, látogass el az alábbi weboldalra, ahol egy ládarendező játékot találsz:

   http://anim.ide.sk/ladakrendezese.php

2. Mielőtt elkezdenél játszani, kérjük, írd be az alábbi mezőbe az azonosítószámod (id), melyet a játék alatti sorban találsz:

id:

   Kérjük, hogy a böngészőt ez után NE zárd be mindaddig, amíg nem töltötted ki a teljes kérdőívet!

3. Most olvasd el figyelmesen, mi a feladat!

   A játékban látható ládák 1, 2, 3 kg súlyúak, de nem tudod melyik milyen nehéz. Hasonlítsd össze a ládák súlyát a kétkarú mérleg segítségével és akaszd fel a ládákat a kampókra (a legkönnyebbtől a legnehezebbig), de vigyázz! Soha ne tegyél egyik kampóra se nehezebb ládát, mint amilyent elbír, különben leszakad!

   A játék három szintből áll: egy 3 ládás, egy 5 ládás, majd egy 7 ládás szintből.

   A játékot próbáld meg minél kevesebb méréssel megoldani! Tehát felesleges mérések nélkül, pl. ne mérd le kétszer ugyanazokat a ládákat, vagy azokat, melyek súlyára esetleg már lehet következtetni az előtte levő mérésekből.

   A játékot bármennyiszer újraindíthatod, sikertelen játék esetén lépésenként visszanézheted és átgondolhatod, hogy mit rontottál el.

   ☺ KEZDJ EL JÁTSZANI! ☺

   (A kérdőv maradék részét csak az után töltsd ki, miután eleget játszottál és rájöttél arra, hogyan lehet a lehető legkevesebb méréssel megoldani a feladatot, vagy már nincs több időd a játékra.)

4. Kérjük, válaszolj az alábbi, játékkal kapcsolatos kérdésekre, a megfelelő pontszám bekarikázásával a számskálán:

   Összességében mennyire tetszett a játék?

   egyáltalán nem tetszett 1 2 3 4 5 6 7 8 9 10 *nagyon tetszett*

   A feladat érthető, világos volt?

   egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, nagyon*

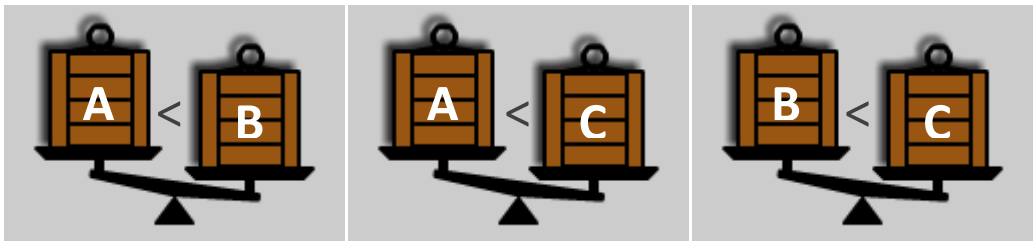
   A játék kezelése könnyen elsajátítható volt?

   egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, nagyon*

   A játék grafikailag megfelelő volt?

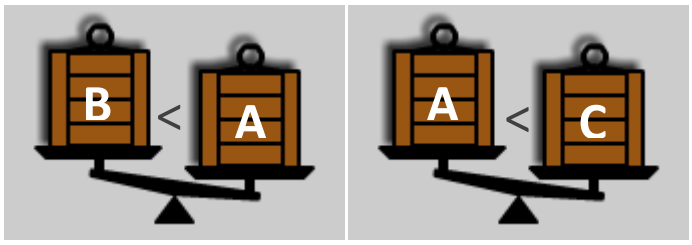
   egyáltalán nem 1 2 3 4 5 6 7 8 9 10 *igen, nagyon*

5. Amennyiben az alábbi méréseket végeztük volna, milyen lenne a ládák sorrendje?



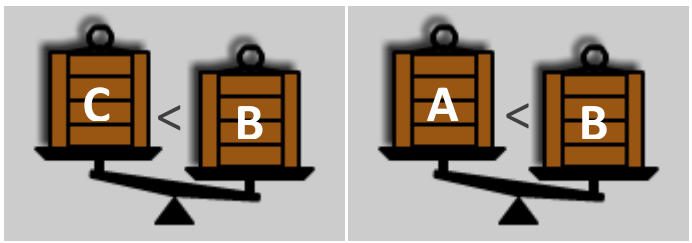
- ☐ **A < C < B**
- ☐ **A < B < C**
- ☐ **B < A < C**
- ☐ **egyéb:** ……
- ☐ **ennyi mérésből nem határozható meg a pontos sorrend**

6. Amennyiben az alábbi méréseket végeztük volna, milyen lenne a ládák sorrendje?



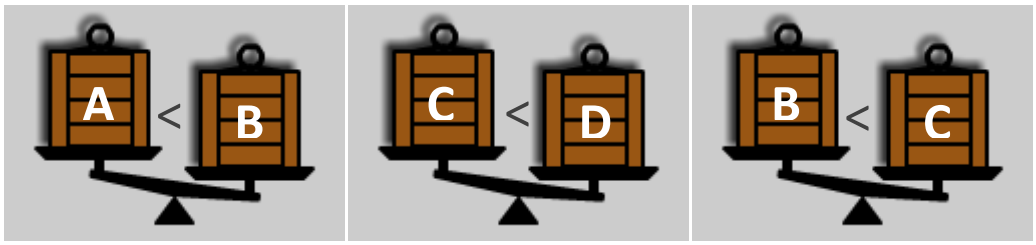
- ☐ **B < C < A**
- ☐ **A < C < B**
- ☐ **B < A < C**
- ☐ **egyéb:** ……
- ☐ **ennyi mérésből nem határozható meg a pontos sorrend**

7. Amennyiben az alábbi méréseket végeztük volna, milyen a lenne ládák sorrendje?



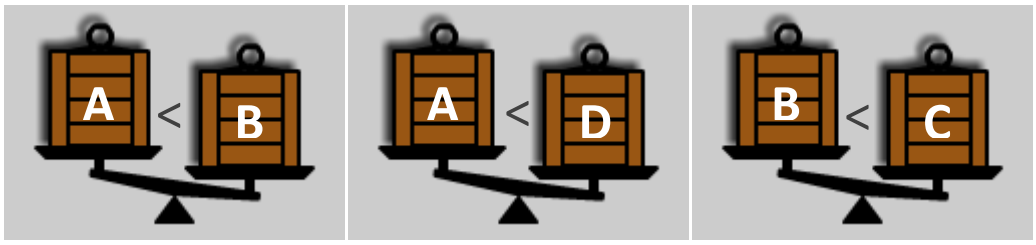
- ☐ **A < B < C**
- ☐ **C < A < B**
- ☐ **A < C < B**
- ☐ **egyéb:** ……
- ☐ **ennyi mérésből nem határozható meg a pontos sorrend**

8. Amennyiben az alábbi méréseket végeztük volna, milyen lenne a ládák sorrendje?



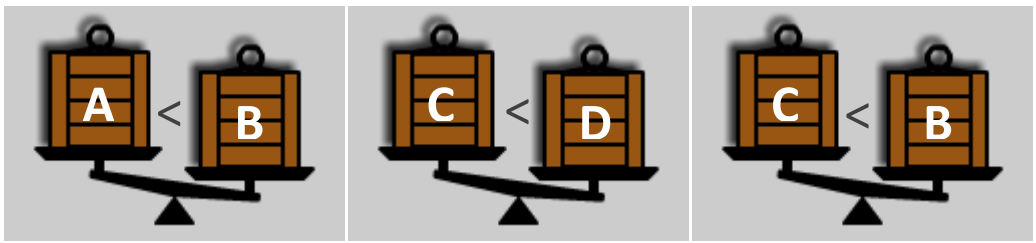
- ☐ **A < B < C < D**
- ☐ **B < C < A < D**
- ☐ **A < B < D < C**
- ☐ **egyéb:** ……
- ☐ **ennyi mérésből nem határozható meg a pontos sorrend**

9. Amennyiben az alábbi méréseket végeztük volna, milyen lenne a ládák sorrendje?



- ☐ **A < B < C < D**
- ☐ **A < D < B < C**
- ☐ **A < B < D < C**
- ☐ **egyéb:** ……
- ☐ **ennyi mérésből nem határozható meg a pontos sorrend**

10. Amennyiben az alábbi méréseket végeztük volna, milyen lenne a ládák sorrendje?



- ☐ **A < B < C < D**
- ☐ **A < C < B < D**
- ☐ **A < C < D < B**
- ☐ **egyéb:** ……
- ☐ **ennyi mérésből nem határozható meg a pontos sorrend**

11. Volt bármi a játékban, ami zavaró volt? Ha igen, írd le röviden mi volt az!

12. Kérjük, most írd le saját szavaiddal, minél érthetőbben a megoldás teljes menetét (algoritmust), ami alapján próbáltad megoldani az 5 vagy 7 ládából álló szintet a számítógépen. Ha sikerült a feladatot felesleges mérések nélkül megoldani, írd le minél pontosabban, hogy hogyan! Nyugodtan játszhatsz közben még egyszer a játékkal, amennyiben szükséged van rá a megoldás pontos átgondolásához.

    Próbáld meg úgy leírni a megoldás menetét, mintha egy olyan diáktársadnak szeretnéd elmagyarázni, akinek nem sikerült teljesítenie a feladatot!

13. melléklet: A ládarendezéssel kapcsolatos kérdőívre adott válaszok és a hozzájuk tartozó adatbázisból kinyert adatok

	kérdőívre adott válaszok											adatbázisból kinyert adatok																	
	a didaktikai játék értékelése				a feladatokra adott válaszok							redundáns (felesleges) mérések száma az első szinten (3 láda)						redundáns (felesleges) mérések száma a második szinten (5 láda)						redundáns (felesleges) mérések száma a harmadik szinten (7 láda)					
diák sorszáma	összességében	érthetőség	kezelhetőség	grafika (szemléletesség)	a kérdőív 5. kérdése	a kérdőív 6. kérdése	a kérdőív 7. kérdése	a kérdőív 8. kérdése	a kérdőív 9. kérdése	a kérdőív 10. kérdése	használt algoritmus a diák által leírt folyamat alapján (a kérdőív 12. kérdése) *	1. sikeres próbálkozásnál	2. sikeres próbálkozásnál	3. sikeres próbálkozásnál	4. sikeres próbálkozásnál	5. sikeres próbálkozásnál	6. sikeres próbálkozásnál	1. sikeres próbálkozásnál	2. sikeres próbálkozásnál	3. sikeres próbálkozásnál	4. sikeres próbálkozásnál	5. sikeres próbálkozásnál	6. sikeres próbálkozásnál	1. sikeres próbálkozásnál	2. sikeres próbálkozásnál	3. sikeres próbálkozásnál	4. sikeres próbálkozásnál	5. sikeres próbálkozásnál	6. sikeres próbálkozásnál
1.	8	10	10	10	b	c	e	a	c	e	bubblesort	0	0	0	0	0	1	5	0	1	4	0	1	9	4	9	9	2	0
2.	9	10	10	9	a	c	b	a	e	e	quicksort	0	0	0	0			0	0	0	0			4	0	0	0		
3.	10	10	10	8	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	4	1
4.	9	9	10	10	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	3	0	0	0	0	0	9	0	0	0	0	
5.	5	10	10	10	b	e	b	d	e	e	bubblesort	1	0	1	0	0	0	5	5	3	0	3	7	14	11	10	15	8	
6.	8	10	9	8	b	c	e	a	e	e	quicksort	1	0	0	0	0		2	0	0	0	0		0	0	0	0	0	
7.	7	9	8	7	b	c	e	a	e	e	bubblesort+	1	0	0	0	0		1	0	0	0			3	0	0	6		
8.	8	9	9	9	b	c	e	a	a	b	bubblesort+	0	0	0	0	0	0	3	2	1	4	0	0	14	6	0	5	0	0
9.	8	9	10	9	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
10.	8	10	10	8	b	c	e	a	e	e	bubblesort+	0	0	1	0	0	0	0	0	6	0	0	0	2	14	8	0	0	2
11.	9	10	10	10	b	c	c	a	e	e	bubblesort+	0	0	0	0	0	0	0	2	0	4	1	0	8	4	3	4	11	0
12.	9	10	10	9	b	c	e	a	e	b	bubblesort+	0	0	2	0	0		5	0	0	0	0		3	2	0	0		
13.					b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	6	0	0	0	0	0	10	0	2	3	0	0
14.		10	10	10	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	2	2	0	4	1	3	9	0	0	0	0	0
15.	7	10	10	3	b	c	e	a	e	b		0	0	0	0	0	0	0	0	3	0	3	1	1	2	2	3	4	2
16.	10	10	4	10	b	c	e	a	a	b		0	0	1	0	1	0	6	3	4	3	3	1	19	15	14	10	0	
17.	6	10	8	10	b	c	e	a	e	c		0	0	0	0	0		0	3	0	0	2		12	1	4	0	0	
18.	7	10	10	7	b	c	e	a	e	e	bubblesort+	0	0	0	0	1	0	0	1	4	0	0	0	4	4	6	0	6	2
19.	8	10	9	10	b	c	e	a	e	e	bubblesort+	0	0	0	0			0	0	0	0			1	0	1			
20.	9	10	9	8	b	c	e	a	e	b	bubblesort+	0	1	1	0	0	0	1	4	1	0	0	0	7	4	1	0	0	0
21.	10	10	10	10	b	c	e	a	e	e	bubblesort+	3	0	0	0	0	0	11	0	0	0	1	0	5	9	0	1	0	1
22.	8	9	10	8	b	c	e	a	e	e	bubblesort+	0	0	0				0	0	0				11	0	0			
23.	9	10	10	10	b	c	e	a	e	e		1	0	0	0	0	0	0	2	1	3	1	5	6	13	8	6	4	13
24.	10	8	8	9	b	c	e	a	e	e	bubblesort+	7	0	0	0	0		4	0	0	1	0		0	1	2	2	0	
25.	8	9	8	7	b	c	e	a	e	e	bubblesort+	0	0	0				4	1	0				9	0	0			
26.	9	10	10	9	b	c	e	a	e	e		0	0	0	1	1	0	0	3	2	2	2	4	3	14	11	9	11	11
27.	9	10	8	9	b	c	b	a	c	b	bubblesort+	1	0	0	0	0	0	5	4	4	5	4	4	14	12	4	10	13	11
28.	8	10	10	9	b	c	e	a	e	c	bubblesort	0	1	1	1	0	0	1	1	6	1	2	1	8	10	9	4	8	0
29.	10	10	10	10	b	c	e	a	b	e	bubblesort+	1	0	0	0	0	0	1	0	0	1	0	1	9	4	9	0	0	1
30.	10	10	10	10	b	c	e	a	a	e	quicksort	0	0	0	0	0	0	3	0	0	0	1	0	10	4	0	0	0	0
31.	5	5	10	10	b	c	e	a	e	e		0	0	0	0	0	0	5	1	0	0	0	0	13	2	3	0	0	
32.	8	10	5	8	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	0	5	1	0	0	0	6	4	0	4	0	0
33.	9	10	10	8	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	2	0	0	0	0	0	0	0	4	1	0	0
34.	5	8	8	9	b	c	e	a	e	b		0	0	3	0	1	0	1	1	3	0	1	1	2	4	1	3	3	6

35.	9	9	9	9	b	c	e	a	e	e	bubblesort	1	0	0	0	0	0	1	2	2	2	0	0	5	5	9	4	2	2
36.	9	10	9	9	b	c	e	a	e	e	bubblesort+	2	1	0	0	0	0	4	1	0	0	0	0	12	4	1	1	0	0
37.	8	10	10	10	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	0	3	0	3	0	0	8	2	0	0	0	2
38.	9	8	8	9	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0
39.	10	10	10	10	b	c	e	a	e	e	bubblesort	1	0	0	0	0	0	4	1	0	1	4	1	9	3	0	6	1	3
40.	10	10	10	9	b	c	e	a	e	e	quicksort	0	0	0	0	0	0	1	2	1	0	0	0	0	0	1	0	0	0
41.	10	10	10	10	b	c	e	a	e	e	quicksort	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1
42.	10	10	10	10	c	c	e	a	e	e	bubblesort+	1	0	0	0			0	0	0	0			1	0	0	0		
43.	9	10	10	10	b	c	e	a	e	e	quicksort	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0
44.	9	10	10	10	b	c	e	a	e	e	quicksort	0	0	0	0	1	0	1	0	1	2	1	0	0	0	4	3	0	1
45.	7	10	10	10	b	c	e	a	e	e		0	0	0				4	2	2				11	0				
46.	10	10	10	8	b	c	c	a	a	b	quicksort	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	
47.	8	10	10	8	b	c	e	a	e	e	quicksort	1	0	0	0	0		1	0	0	0			3	0	0	0		
48.	8	10	10	8	b	c	e	a	a	e		0	0	0	0	0	0	5	2	5	1	1	0	5	3	14	11	13	3
49.	8	10	10	9	b	c	e	a	e	e		0	0	0				0						0					
50.	9	10	10	10	b	c	e	a	e	e		0	0	0	0	0	0	2	1	3	2	0		2	3	4	3		
51.	9	10	10	9	b	c	e	a	e	e	quicksort	0	0	0	0	0	0	4	0	0	0	0	0	13	1	0	0	0	0
52.	9	10	10	10	b	c	e	a	e	e		0	0	0	0	0	0	1	1	0	0	2	3	8	3	4	5	3	
53.	10	10	10	10	b	c	e	a	e	e		0	0	0	0	0	0	5	5	4	0	0	1	4	11	8	3	0	5
54.					b	c	e	a	e	e		0	1	0	0	0	0	5	2	0	2	0		7	8	7	6	10	
55.	10	10	10	10	b	c	e	a	e	e		0	0	0	0	0		9	10	5	1	1		15	17	2	10		
56.	10	10	10	10	b	c	e	a	e	e	quicksort	0	0	0	0	0	0	1	0	2	0	0	0	2	1	1	0	0	0
57.	9	8	9	9	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	1	1	0	0	1	1	0	1	8	5	2	1	0
58.	9	10	10	10	b	c	e	a	e	e	bubblesort	3	0	0	0	0	0	6	4	3	5	4	0	9	11	12	14	14	12
59.	9	5	5	5	b	c	e	a	c	a	bubblesort	2	1	1	0	0	0	17	7	1	0	3	1	32	17	7	4	6	1
60.	10	8	9	8	b	a	e				bubblesort+	0	0	0	0	0	1	0	0	0	0	0	0	6	13	7	1	2	1
61.	9	8	9	9	b	c	e	a	e	e	quicksort	0	1	1	1	0		5	3	5	3	0		11	5	0			
62.	10	10	10	10	b	c	e	a	e	e	mergesort3	1	0	0	0	1	1	4	2	0	0	1	0	17	2	5	1	0	1
63.					b	a	c	a	a	b	bubblesort+	3	0	0	0	0	0	1	1	0	1	0	0	7	5	1	0	0	1
64.	7	10	10	10	c	c	e	a	b	a		0	0	0	0			4	0					5	7				
65.	10	10	10	10	b	c	e	a	e	b	bubblesort	0	0	0	0	0	0	0	2	2	0	0	1	4	3	7	2	5	10
66.	10	8	10	10	b	c	e	a	e	e	bubblesort	1	0	0	0	1	0	0	1	2	4	3	5	12	12	9	11	12	2
67.	9	10	10	10	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	6	0	0	0	0	1	7	4	0	2	0	0
68.	10	7	10	10	b	c	b	a	a	b	bubblesort	0	1	0				1	7	0				3	7				
69.	5	10	10	10	b	c	e	a	e	b		3	0	0	0			10	5	2	1			11	10	10			
70.	8	10	7	9	b	c	b	a	e	b	bubblesort	6	1	0	0			2	4	2	2			10	6	15	11		
71.	5	10	10	10	b	c	e	a	e	e	bubblesort	1	0	0	0			3	3	1	3			16	13	13			
72.	10	10	10	10	b	c	c	a	e	b	bubblesort	0	0	0	0	0	0	3	2	3	1	0	0	11	1	4	5	9	2
73.	6	5	10	10	b	c	e	e				0	0	0	0	1	0	1	7	4	8	3	4	16	6	15	22	20	17
74.	10	10	10	7	d	d	e	d	e	e	bubblesort	1	0	0	0	0	0	3	0	0	2	0	0	5	3	4	4	3	
75.	7	7	9	9	b	c	e	a	e	b		1	0	0	0	0	0	0	4	0	1	0	1	3	3	0	4	4	3
76.	10	10	10	10	b	c	e	a	e	e	bubblesort	0	0	0	0	0		1	2	3	0	0		1	8	7	0		
77.	10	10	10	10	b	c	e	a	e	e	bubblesort	1	0	0	0	0	0	1	4	4	0	0	0	11	12	6	7	0	2
78.	10	10	10	10	b	c	e	a	e	e		0	0	0	0	0	0	1	1	0	0	0	0	8	0	0	0	0	0
79.	7	10	8	8	b	c	e	a	e	e	quicksort	0	2	0	0	0	0	1	1	1	1	0	0	11	0	0	0	0	0
80.	10	10	10	10	b	c	e	a	e	e		0	0	0	0	0	0	3	0	0	0	0	0	12	0	1	0	1	4
81.	10	10	10	10	b	c	e	a	e	b	bubblesort	1	0	0	1	1	0	3	7	2				14					
82.	8	8	9	10	b	c	e	a	e	e		5	0	0	0	0	0	4	4	3	3	1		10	6				
83.	4	8	5	4	b	c	e	a	b	c		2	0	0	0			3	6	5	0			9	10	2	3		
84.	8	10	10	10	b	c	b	a	e	e	bubblesort	0	0	0	0	0	0	0	3	0	1	0	0	8	7	0	3	3	3
85.	10	10	10	10	b	c	e	a	e	e		0	0	1	0	0	0	3	4	2	6	1	4	8	14	10	15	13	5
86.	5	10	10	10	b	c	e	a	e	b		2	0	0	0			1	1	1	0			11	4	0			
87.	9	10	10	8	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
88.	6	10	10	10	b	c	e	a	b	e		0	1	1	0			5	4	4				8	14	6			
89.	10	10	10	10	b	c	c	a	a	c		1	0	0	0	0	0	4	1	0	0	1	0	2	3	4	3	1	2

90.	8	10	10	10	b	c	e	a	e	b		1	1	0	0			5	3	0	1			8	11	0			
91.	4	8	9	8	b	c	e	a	e	e	mergesort3	0	0	0	0	0	0	0	0	4	0	0	0	2	1	0	1	1	0
92.	6	10	10	10	b	c	b	a	e	e	mergesort3	0	0	0	0	0	0	2	0	2	1	0	0	8	11	14	2	0	0
93.	10	10	10	10	b	c	e	a	e	e	bubblesort+	0	1	1	0	0	0	1	1	2	2	0	0	11	7	1	4	0	1
94.	10	10	10	7	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	1	1	0
95.	10	10	10	10	b	c	e	a	e	e	mergesort3	0	1	0	0	0	0	0	0	0	0	0		2	0	0	0		
96.	10	10	10	8	b	d	e	d	e	e	bubblesort	3	1	1	0	0	0	6	5	4	0	0	2	8	7	13	1	2	3
97.	10	10	10	10	b	c	e	a	e	e	bubblesort	0	1	0	1	0	0	1	0	0	0	0	1	4	3				
98.												1	0	0	0	0	0	10	4	2	2	4	3	6	3	10	11	9	10
99.	10	10	9	10	b	c	e	a	e	e	bubblesort+	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0

Helyes válaszok:

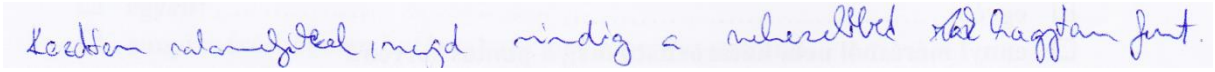
b	c	e	a	e	e

* „bubblesort+“ algoritmussal jelöltük meg azon rendezési algoritmusokat, ahol a diákok egy módosított buborékrendezés alapján rendezték a ládákat. Ez egy buborékrendezésre épülő rendezés, amelyet kiegészítetek ládák megjelölésével (külön csoportokba rakásukkal) annak érdekében, hogy ne végezzenek redundáns (felesleges) méréseket.
* „mergesort3“ algoritmussal jelöltük meg azon rendezési algoritmusokat, ahol a diákok egy módosított összefésülő rendezés alapján rendezték a ládákat. Pl. 7 láda rendezésénél 3-elemű rendezett csoportokat alakítottak ki a diákok, majd az egyik csoporthoz hozzáadták a még kimarató elemet, végül ezt a két csoportot összefésülték.

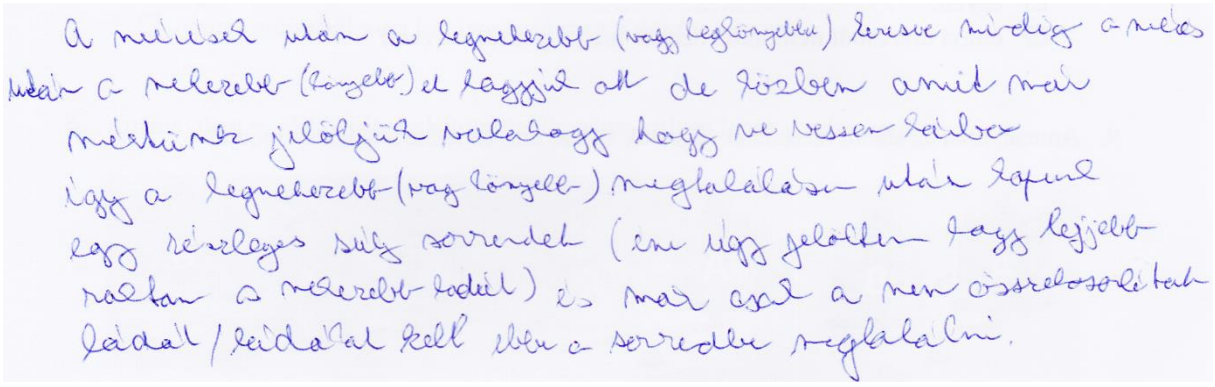
Diákok sorszáma:

- 2014/15-ös évben végzett tesztelések: 1. – 50.
- 2015/16-os évben végzett tesztelések: 51. – 99.

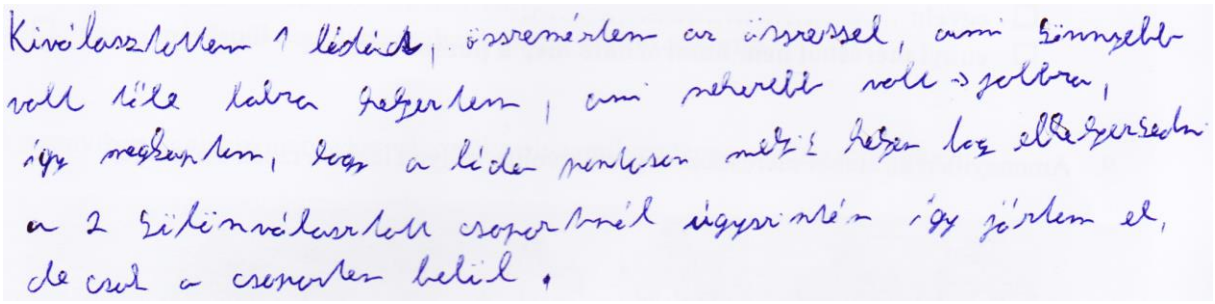
Diák által leírt bubblesort algoritmus (válasz a kérdőív 12. kérdésére):



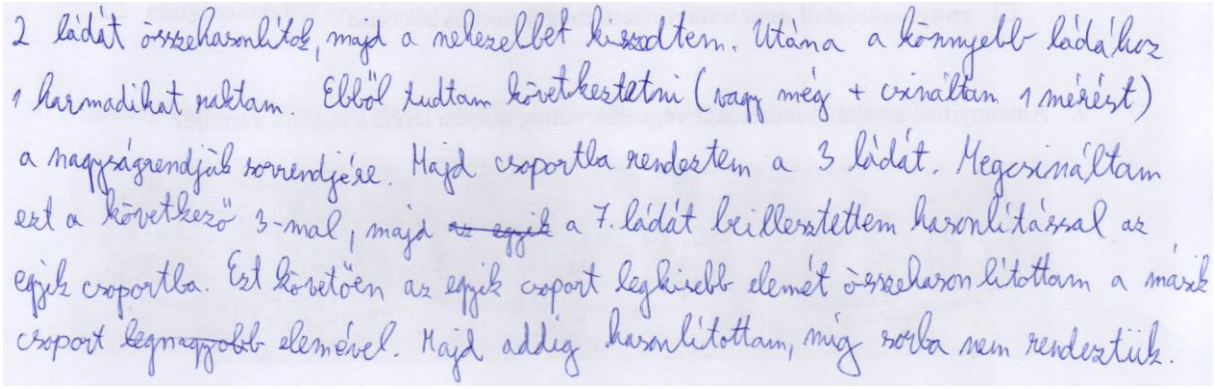
Diák által leírt bubblesort+ algoritmus (válasz a kérdőív 12. kérdésére):



Diák által leírt quicksort algoritmus (válasz a kérdőív 12. kérdésére):


Kiválasztottam 1 ládát, összemértem az összessel, ami könnyebb volt tőle balra helyeztem, ami nehezebb volt → jobbra, így megtudtam, hogy a láda pontosan melyik helyen lesz elhelyezkedni a 2 kiválasztott csoportnál ugyanúgy így járok el, de csak a csoporton belül.

Diák által leírt mergesort3 algoritmus (válasz a kérdőív 12. kérdésére):


2 ládát összehasonlítok, majd a nehezebbet kiszedtem. Utána a könnyebb ládához 1 harmadikat raktam. Ebből tudtam következtetni (vagy még + csináltam 1 mérést) a nagyságrendjük sorrendjére. Majd csoportba rendeztem a 3 ládát. Megcsináltam ezt a következő 3-mal, majd ~~az egyik~~ a 7. ládát beillesztettem hasonlítással az egyik csoportba. Ezt követően az egyik csoport legkisebb elemét összehasonlítottam a másik csoport legnagyobb elemével. Majd addig hasonlítottam, míg sorba nem rendeztük.

14. melléklet: Különféle buborékrendező algoritmusok és a gyorsrendező algoritmus Java programja

```
import java.util.Random;

public class App {

	private static int  n = 10000;
	private static int[] org, a;
	private static long c, e;

	public static void main(String[] args) {

		org = new int[n];
		Random rgen = new Random();
		for (int i=0; i<n; i++) {
			org[i] = rgen.nextInt();
		}
		a = new int[n];

		long start, stop;

		c = 0; e = 0;
		System.arraycopy(org, 0, a, 0, n);
		start = System.currentTimeMillis();
		bubblesort1();
		stop = System.currentTimeMillis();
		System.out.println(c + " összehasonlítás, " + e + " csere, "
		+ (stop-start) + " ms");

		c = 0; e = 0;
		System.arraycopy(org, 0, a, 0, n);
		start = System.currentTimeMillis();
		bubblesort2();
		stop = System.currentTimeMillis();
		System.out.println(c + " összehasonlítás, " + e + " csere, "
		+ (stop-start) + " ms");

		c = 0; e = 0;
		System.arraycopy(org, 0, a, 0, n);
		start = System.currentTimeMillis();
		bubblesort3(0, n-1);
		stop = System.currentTimeMillis();
		System.out.println(c + " összehasonlítás, " + e + " csere, "
		+ (stop-start) + " ms");

		c = 0; e = 0;
		System.arraycopy(org, 0, a, 0, n);
		start = System.currentTimeMillis();
		quicksort(0, n-1);
		stop = System.currentTimeMillis();
		System.out.println(c + " összehasonlítás, " + e + " csere, "
		+ (stop-start) + " ms");

	}
```

```
// hagyományos buborékrendezés
private static void bubblesort1() {
	for (int i=n-1; i>0; i--) {
		for (int j=0; j<i; j++) {
			c++;
			if (a[j]>a[j+1]) exchange(j, j+1);
		}
	}
}

// továbbfejlesztett buborékrendezés
private static void bubblesort2() {
	int k, i = n - 1;
	while (i>0) {
		k = 0;
		for (int j=0; j<i; j++) {
			c++;
			if (a[j]>a[j+1]) {
				exchange(j, j+1);
				k = j;
			}
		}
		i = k;
	}
}

// diákok által kigondolt buborekrendezés (rekurzív változat)
private static void bubblesort3(int low, int high) {
	int i, k = 0;
	for (i=low; i<high; i++) {
		c++;
		if (a[i]>a[i+1]) {
			exchange(i, i+1);
			k++;
		} else {
			int newLow = i-1-k < 0 ? 0 : i-1-k;
			int newHigh = i-1;
			if (newLow<newHigh) bubblesort3(newLow, newHigh);
			k = 0;
		}
	}
	int newLow = i-1-k < 0 ? 0 : i-1-k;
	int newHigh = i-1;
	if (newLow<newHigh) bubblesort3(newLow, newHigh);
}

// gyorsrendezés
private static void quicksort(int low, int high) {
    int i = low, j = high;
    int pivot = a[low + (high-low)/2];
    while (i <= j) {
      while (a[i] < pivot) {
		c++;
		i++;
	}
	while (a[j] > pivot) {
		c++;
		j--;
	}
```

```
            if (i <= j) {
                  exchange(i, j);
                  i++;
                  j--;
            }
        }
            if (low < j)
                quicksort(low, j);
            if (i < high)
                quicksort(i, high);
    }

    // elemek cseréjéhez használt függvény
    private static void exchange(int i, int j) {
        int temp = a[i];
        a[i] = a[j];
        a[j] = temp;
        e++;
    }

}
```

15. melléklet: Dinamikus adatszerkezetekkel kapcsolatos kérdőív (Google űrlap)

Egyirányú láncolt lista kialakítása

Az alkalmazás itt érhető el: http://anim.ide.sk/lista.php

Kérjük értékeld az alkalmazást, minden egyes kérdést alaposan gondolj át, ha szükséges, többször olvass el. Őszinte válaszaiddal segítheted hasonló alkalmazások létrejöttét és az oktatás színvonalának növelését. Köszönjük!

1. Határozd meg alkalmazás használatának HATÉKONYSÁGÁT AZ OKTATÁSBAN!

egyáltalán nem hatékony 1 2 3 4 5 nagyon hatékony

2. Segített az alkalmazás az EGYIRÁNYÚ LÁNCOLT LISTA fogalmának megértésében?

egyáltalán nem 1 2 3 4 5 igen, nagyon

3. Mennyire segített az alkalmazás az egyirányú láncolt listát használó PASCAL PROGRAM MEGÍRÁSÁNÁL?

egyáltalán nem 1 2 3 4 5 nagyon segített

4. Határozd meg, mennyire FELHASZNÁLÓBARÁT, mennyire könnyen kezelhető az alkalmazás!

egyáltalán nem 1 2 3 4 5 nagyon felhasználóbarát

5. Segítettek az alkalmazásban található PIROS NYOMÓGOMBOK (PASCAL PARANCSOK) a dinamikus adatszerkezetek megértésében?

egyáltalán nem 1 2 3 4 5 igen, nagyon

6. Segítettek az alkalmazásban az UTOLJÁRA HASZNÁLT PASCAL PARANCSOK LISTÁJA (a használat során megjelenő piros gombok a jobb oldalon) a dinamikus adatszerkezetek megértésében?

egyáltalán nem 1 2 3 4 5 igen, nagyon

7. Segítettek az alkalmazásban a VEZÉRLŐGOMBOK (jobb oldalon található kör alakú gombok - VISSZALÉPÉS, ELŐRE LÉPÉS, LISTA ÁTRENDEZÉSE, SÚGÓ, stb.) a dinamikus adatszerkezetek megértésében?

egyáltalán nem 1 2 3 4 5 igen, nagyon

8. Mennyire SZEMLÉLETES az alkalmazásban a láncolt lista ÁBRÁZOLÁSA a téglalapok és a nyilak segítségével?

egyáltalán nem szemléletes 1 2 3 4 5 nagyon szemléletes

9. Szerinted elegendő az INTERAKTIVITÁS az alkalmazásban (ahogy az alkalmazás reagál a gombnyomásokra)?

egyáltalán nem 1 2 3 4 5 igen, teljes mértékben

10. Javasolnád az ilyen oktatási segédeszközök használatát az oktatásban?

egyáltalán nem 1 2 3 4 5 igen, nagyon

11. Mit változtatnál az alkalmazásban, mivel egészítenéd ki, mit hagynál ki belőle? (fogalmazd meg röviden, pár mondatban)

.....

12. Az alkalmazásban található Pascal utasítások készlete elégséges? Hiányzik valamilyen utasítás?

.....

16. melléklet: Az „inalan” könyvtár segítségével kialakított animáció forráskódjai

tomb_tukrozese.html

```
<!DOCTYPE html>
<head>
   <title>Tömb tükrözése</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <style>
            canvas {
                  border: 1px solid black;
            }
      </style>
      <script src="http://inalan.ide.sk/inalan20150819.js"></script>
      <script src="tomb_tukrozese.js"></script>
</head>
<body onload="init();">
      <canvas id="myCanvas" width="700" height="440"></canvas>
</body>
</html>
```

tomb_tukrozese.js

```
var init = function () {

      var stage = new inalan.Stage("myCanvas");
      stage.showAllButtons();

      var a = new inalan.VisuArray("a", [0, 0, 0, 0, 0, 0, 0, 0], true);
      a.setMinValue(5);
      a.randomize(30, 150);
      a.x = 80;
      a.y = 220;
      stage.add(a, "a");

      var c = new inalan.VisuCode( [ "CIKLUS i = 0-tól 3-ig",
                                     "   csere a[i] a[7-i]" ] );
      c.x = 360;
      c.y = 80;
      stage.add(c, "code");


      stage.controller.startLabel = "Lejátszás";
      stage.controller.stopLabel = "Leállítás";
      stage.controller.speedLabel = "Az animáció sebessége:";
      stage.controller.startStop.text = stage.controller.startLabel;
      stage.controller.startStop.width = 80;
      stage.controller.speed.text = stage.controller.speedLabel;

      // *****************************************
      // az animáció lépéseinek definiálása...

      stage.vars.i = 0;

      var line0 = function () {
            stage.get("code").selected = [0];
            stage.get("a").setIndex("i", stage.vars.i);
```

```
            stage.get("a").setLoopMarker("i", 0, 3);
            return 200;
        }

        var line1 = function () {
            stage.get("code").selected = [1];
            stage.get("a").setIndex("7-i", 7 - stage.vars.i);
            stage.swap( stage.get("a")[stage.vars.i],
                        stage.get("a")[7-stage.vars.i] );
        }

        var incI = function () {
            stage.get("a").deleteIndex("7-i");
            stage.get("a")[stage.vars.i].setGreenColor();
            stage.get("a")[7-stage.vars.i].setGreenColor();
            stage.vars.i++;
            return 0;
        }

        var checkI = function () {
            return stage.vars.i <= 3;
        }

        var finalStep = function () {
            stage.get("code").selected = [];
            stage.get("a").deleteIndex("i");
            stage.get("a").deleteIndex("7-i");
        }

        // ********************************************
        // a lépések sorrendjének definiálása...

        stage.setSteps([
            [
              line0,
              line1,
              incI
            ], checkI,
            finalStep
        ]);

}
```

17. melléklet: Előzetes tudás felmérésére szolgáló kérdőív (mindkét csoport)

1. Mennyi az y értéke a feltételvizsgálat után? ☐ 0 ☐ 3 ☐ 5 ☐ 8 ☐ 11

```
int y=0;
if (y<5) {
     y=y+8;
} else {
     y=y+3
}
```

2. Hányszor írja ki az alábbi program a HELLO szöveget? ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10

```
for (int i=0; i<7; i++) {
     printf("HELLO");
}
```

3. Hányszor írja ki az alábbi program a HELLO szöveget? ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 7

```
for (int i=0; i<7; i++) {
     if (i>2) {
          printf("HELLO");
     }
}
```

4. Mennyi az x értéke a ciklus lefutása után? ☐ 0 ☐ 2 ☐ 4 ☐ 5 ☐ 6

```
int x=0;
while (x<5) {
     x=x+2;
}
```

5. Mennyi az x értéke a ciklus lefutása után? ☐ 0 ☐ 2 ☐ 4 ☐ 5 ☐ 6

```
int x=0;
do {
     x=x+2;
} while (x<5);
```

6. Mennyi az x értéke a ciklus lefutása után? ☐ 0 ☐ 3 ☐ 4 ☐ 5 ☐ 8

```
int x=0;
for (int i=8; i>3; i--) {
     x++;
}
```

7. Mennyi az x értéke a ciklus lefutása után? ☐ 0 ☐ 2 ☐ 3 ☐ 4 ☐ 5

```
int x=0;
for (int i=0; i<5; i++) {
     x=i;
}
```

8. Mennyi az x értéke a ciklusok lefutása után? ☐ 0 ☐ 2 ☐ 3 ☐ 5 ☐ 6

```
int x=0;
for (int i=1; i<=3; i++) {
     for (int j=1; j<=2; j++) {
          x++;
     }
}
```

9. Mennyi a tömbelemek értéke a ciklus lefutása után?

```
int a[5];
for (int i=0; i<5; i++) {
     a[i]=1;
}
```

☐ 0, 0, 0, 0, 0
☐ 1, 1, 1, 1, 0
☐ 0, 1, 1, 1, 0
☐ 0, 1, 1, 1, 1
☐ 1, 1, 1, 1, 1

10. Mennyi a tömbelemek értéke a ciklus lefutása után?

```
int a[5];
for (int i=0; i<5; i++) {
     a[i]=i*2;
}
```

☐ 0, 1, 2, 3, 4
☐ 0, 2, 4, 6, 0
☐ 0, 2, 4, 6, 8
☐ 2, 4, 6, 8, 0
☐ 2, 4, 6, 8, 10

11. Mennyi a tömbelemek értéke a ciklus lefutása után?

```
int a[5];
for (int i=0; i<5; i++) {
     if (i<=2) {
          a[i]=1;
     } else {
          a[i]=2;
     }
}
```

☐ 0, 0, 0, 0, 0
☐ 0, 0, 0, 1, 2
☐ 0, 1, 1, 2, 0
☐ 0, 1, 1, 2, 2
☐ 1, 1, 1, 2, 2
☐ 1, 1, 2, 2, 2
☐ 2, 2, 1, 1, 1
☐ 2, 2, 2, 1, 1

12. Mennyi a tömbelemek értéke a ciklus lefutása után?

```
int a[5];
for (int i=0; i<5; i++) {
     a[i]=i;
     for(int j=0; j<=2; j++) {
          a[i]++;
     }
}
```

☐ 0, 0, 0, 0, 0
☐ 0, 1, 2, 3, 0
☐ 0, 1, 2, 3, 4
☐ 1, 2, 3, 4, 5
☐ 2, 3, 4, 5, 6
☐ 3, 4, 5, 6, 7
☐ 4, 5, 6, 7, 8
☐ 4, 5, 6, 7, 0

18. melléklet: Egyszerű cserés rendezéssel kapcsolatos kérdőív (1. csoport - animáció)

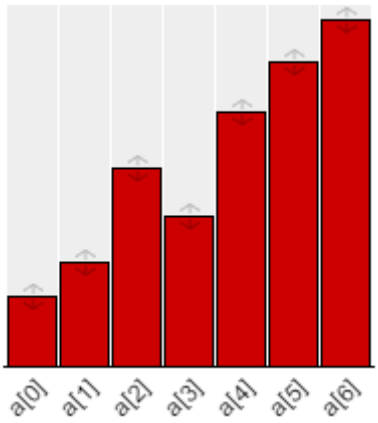
1. Az alábbiak közül mi igaz az **egyszerű cserés rendezésre (simplesort)**? Több válasz is bejelölhető!

- ☐ az elemeket összehasonlítja az összes mögötte levő elemmel
- ☐ csak a közvetlenül egymás melletti elemeket hasonlítja össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

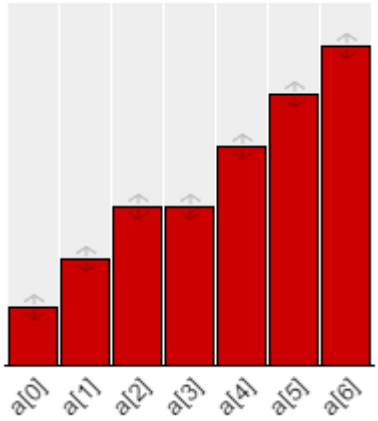
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** van amikor kisebb, van amikor nagyobb mint **j**
- ☐ az **i** van amikor kisebb, van amikor egyenlő **j**-vel, de sose nagyobb **j**-nél
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 6
- ☐ 11
- ☐ 15
- ☐ 21

4. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 6
- ☐ 11
- ☐ 15
- ☐ 21

5. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=0; i<=5; i++) {
     for (int j=i+1; j<=6; j++) {
          if (a[i]>a[j]) {
               csere(&a[i],&a[j]);
          }
     }
}
```

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=0; i<=5; i++) {
     for (int j=i+1; j<=6; j++) {
          if (a[i]>a[j]) {
               csere(&a[i],&a[j]);
          }
     }
}
```

19. melléklet: Egyszerű cserés rendezéssel kapcsolatos kérdőív (2. csoport - grafika)

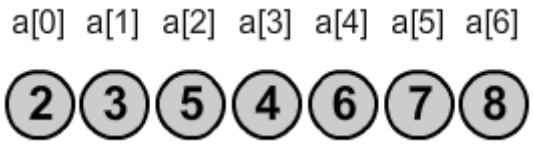
1. Az alábbiak közül mi igaz az **egyszerű cserés rendezésre (simplesort)**? Több válasz is bejelölhető!

- ☐ az elemeket összehasonlítja az összes mögötte levő elemmel
- ☐ csak a közvetlenül egymás melletti elemeket hasonlítja össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

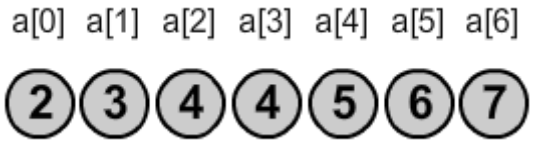
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** van amikor kisebb, van amikor nagyobb mint **j**
- ☐ az **i** van amikor kisebb, van amikor egyenlő **j**-vel, de sose nagyobb **j**-nél
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 6
- ☐ 11
- ☐ 15
- ☐ 21

4. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 6
- ☐ 11
- ☐ 15
- ☐ 21

5. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=0; i<=5; i++) {
     for (int j=i+1; j<=6; j++) {
          if (a[i]>a[j]) {
               csere(&a[i],&a[j]);
          }
     }
}
```

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=0; i<=5; i++) {
     for (int j=i+1; j<=6; j++) {
          if (a[i]>a[j]) {
               csere(&a[i],&a[j]);
          }
     }
}
```

20. melléklet: Buborékrendezéssel kapcsolatos kérdőív (1. csoport - animáció)

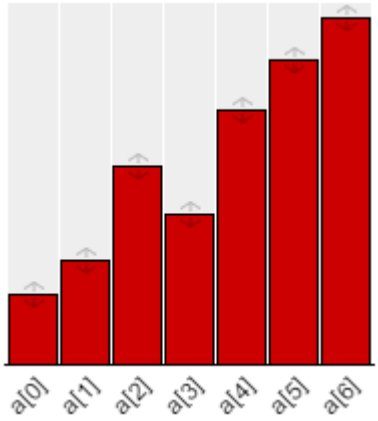
1. Az alábbiak közül mi igaz a **buborékrendezésre (bubblesort)**? Több válasz is bejelölhető!

- ☐ egymástól távolabb elhelyezkedő elemeket is összehasonlít
- ☐ közvetlenül egymás melletti (szomszédos) elemeket hasonlít össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

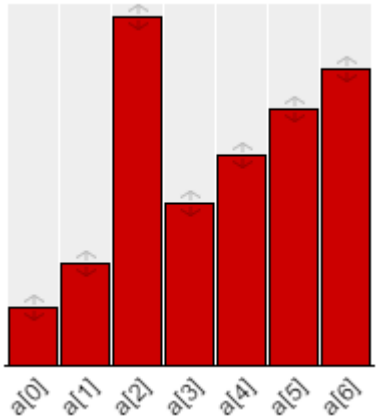
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** van amikor kisebb, van amikor nagyobb mint **j**
- ☐ az **i** van amikor kisebb, van amikor egyenlő **j**-vel, de sose nagyobb **j**-nél
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



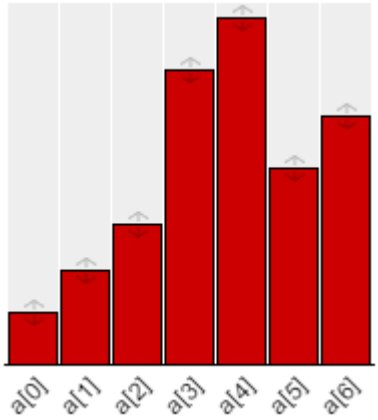
- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

4. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

5. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=6; i>0; i--) {
     for (int j=0; j<i; j++) {
          if (a[j]>a[j+1]) {
               csere(&a[j],&a[j+1]);
          }
     }
}
```

7. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=6; i>0; i--) {
     for (int j=0; j<i; j++) {
          if (a[j]>a[j+1]) {
               csere(&a[j],&a[j+1]);
          }
     }
}
```

21. melléklet: Buborékrendezéssel kapcsolatos kérdőív (2. csoport - grafika)

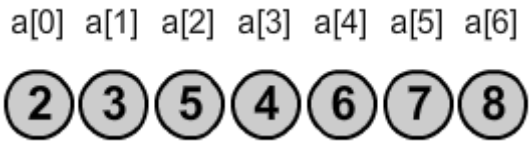
1. Az alábbiak közül mi igaz a **buborékrendezésre (bubblesort)**? Több válasz is bejelölhető!

- ☐ egymástól távolabb elhelyezkedő elemeket is összehasonlít
- ☐ közvetlenül egymás melletti (szomszédos) elemeket hasonlít össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

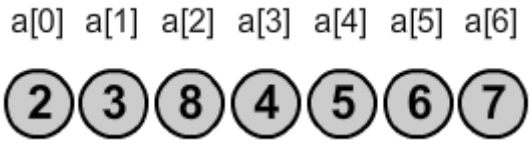
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** van amikor kisebb, van amikor nagyobb mint **j**
- ☐ az **i** van amikor kisebb, van amikor egyenlő **j**-vel, de sose nagyobb **j**-nél
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



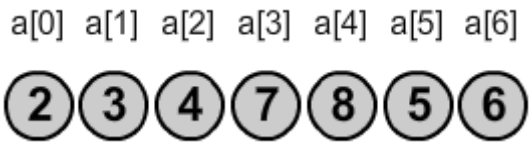
- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

4. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

5. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=6; i>0; i--) {
     for (int j=0; j<i; j++) {
          if (a[j]>a[j+1]) {
               csere(&a[j],&a[j+1]);
          }
     }
}
```

7. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=6; i>0; i--) {
     for (int j=0; j<i; j++) {
          if (a[j]>a[j+1]) {
               csere(&a[j],&a[j+1]);
          }
     }
}
```

22. melléklet: Továbbfejlesztett buborékrendezéssel kapcsolatos kérdőív (1. csoport - animáció)

1. Az alábbiak közül mi igaz a **továbbfejlesztett buborékrendezésre (improved bubblesort)**? Több válasz is bejelölhető!

- ☐ egymástól távolabb elhelyezkedő elemeket is összehasonlít
- ☐ közvetlenül egymás melletti (szomszédos) elemeket hasonlít össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

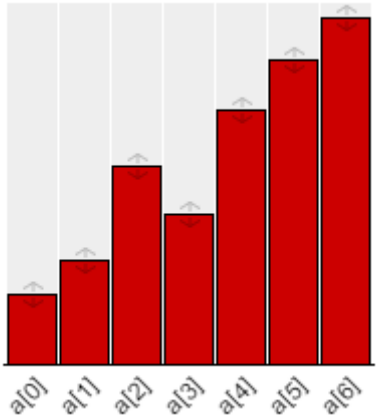
2. Mi igaz az **i** és a **j** változókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** van amikor kisebb, van amikor nagyobb mint **j**
- ☐ az **i** van amikor kisebb, van amikor egyenlő **j**-vel, de sose nagyobb **j**-nél
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Mi igaz az **i** változóra az alábbiak közül? Csak egy válasz helyes!

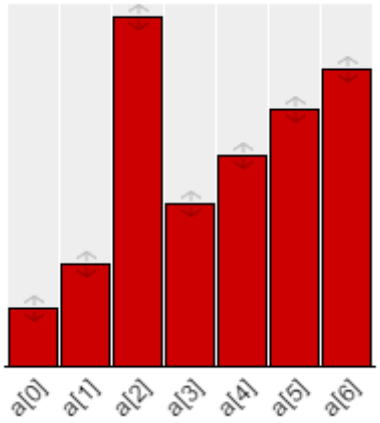
- ☐ az **i** értéke mindig egyesével csökken
- ☐ az **i** értéke mindig egyesével növekszik
- ☐ az **i** értéke mindig csökken, de van hogy több mint eggyel
- ☐ az **i** értéke mindig növekszik, de van hogy több mint eggyel
- ☐ az **i** értéke néha csökken, néha növekszik

4. Mennyi különböző értéket és milyen értékeket fog az **i** változó felvenni az algoritmus futása alatt, ha a rendezés előtt az alábbi tömbünk van?



- ☐ nyolc értéket (i = 6, 5, 4, 3, 2, 1, 0, -1)
- ☐ hét értéket (i = 6, 5, 4, 3, 2, 1, -1)
- ☐ hat értéket (i = 6, 5, 4, 3, 2, -1)
- ☐ öt értéket (i = 6, 3, 2, 1, -1)
- ☐ négy értéket (i = 6, 2, 1, 0)
- ☐ három értéket (i = 2, 1, 0)
- ☐ három értéket (i = 6, 2, -1)
- ☐ három értéket (i = 6, 2, 0)

5. Mennyi különböző értéket és milyen értékeket fog az **i** változó felvenni az algoritmus futása alatt, ha a rendezés előtt az alábbi tömbünk van?



- ☐ nyolc értéket (i = 6, 5, 4, 3, 2, 1, 0, -1)
- ☐ hét értéket (i = 6, 5, 4, 3, 2, 1, -1)
- ☐ hat értéket (i = 6, 5, 4, 3, 2, -1)
- ☐ öt értéket (i = 6, 3, 2, 1, -1)
- ☐ négy értéket (i = 6, 5, 1, 0)
- ☐ három értéket (i = 5, 1, 0)
- ☐ három értéket (i = 6, 5, -1)
- ☐ három értéket (i = 6, 5, 0)

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
int i=6;
while (i>0) {
      int cs=-1;
      for (int j=0; j<i; j++) {
            if (a[j]>a[j+1]) {
                  csere(&a[j],&a[j+1]);
                  cs=j;
            }
      }
      i=cs;
}
```

7. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
int i=6;
while (i>0) {
      int cs=-1;
      for (int j=0; j<i; j++) {
            if (a[j]>a[j+1]) {
                  csere(&a[j],&a[j+1]);
                  cs=j;
            }
      }
      i=cs;
}
```

23. melléklet: Továbbfejlesztett buborékrendezéssel kapcsolatos kérdőív (2. csoport - grafika)

1. Az alábbiak közül mi igaz a **továbbfejlesztett buborékrendezésre (improved bubblesort)**? Több válasz is bejelölhető!

- ☐ egymástól távolabb elhelyezkedő elemeket is összehasonlít
- ☐ közvetlenül egymás melletti (szomszédos) elemeket hasonlít össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

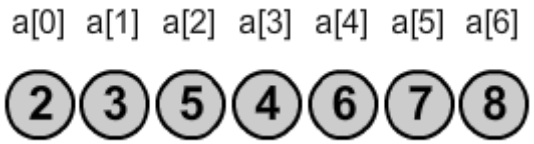
2. Mi igaz az **i** és a **j** változókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** van amikor kisebb, van amikor nagyobb mint **j**
- ☐ az **i** van amikor kisebb, van amikor egyenlő **j**-vel, de sose nagyobb **j**-nél
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Mi igaz az **i** változóra az alábbiak közül? Csak egy válasz helyes!

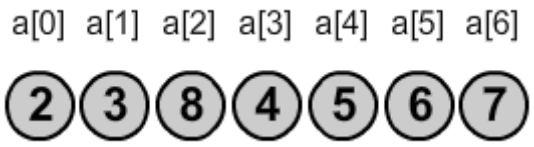
- ☐ az **i** értéke mindig egyesével csökken
- ☐ az **i** értéke mindig egyesével növekszik
- ☐ az **i** értéke mindig csökken, de van hogy több mint eggyel
- ☐ az **i** értéke mindig növekszik, de van hogy több mint eggyel
- ☐ az **i** értéke néha csökken, néha növekszik

4. Mennyi különböző értéket és milyen értékeket fog az **i** változó felvenni az algoritmus futása alatt, ha a rendezés előtt az alábbi tömbünk van?



- ☐ nyolc értéket (i = 6, 5, 4, 3, 2, 1, 0, -1)
- ☐ hét értéket (i = 6, 5, 4, 3, 2, 1, -1)
- ☐ hat értéket (i = 6, 5, 4, 3, 2, -1)
- ☐ öt értéket (i = 6, 3, 2, 1, -1)
- ☐ négy értéket (i = 6, 2, 1, 0)
- ☐ három értéket (i = 2, 1, 0)
- ☐ három értéket (i = 6, 2, -1)
- ☐ három értéket (i = 6, 2, 0)

5. Mennyi különböző értéket és milyen értékeket fog az **i** változó felvenni az algoritmus futása alatt, ha a rendezés előtt az alábbi tömbünk van?



- ☐ nyolc értéket (i = 6, 5, 4, 3, 2, 1, 0, -1)
- ☐ hét értéket (i = 6, 5, 4, 3, 2, 1, -1)
- ☐ hat értéket (i = 6, 5, 4, 3, 2, -1)
- ☐ öt értéket (i = 6, 3, 2, 1, -1)
- ☐ négy értéket (i = 6, 5, 1, 0)
- ☐ három értéket (i = 5, 1, 0)
- ☐ három értéket (i = 6, 5, -1)
- ☐ három értéket (i = 6, 5, 0)

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
int i=6;
while (i>0) {
      int cs=-1;
      for (int j=0; j<i; j++) {
            if (a[j]>a[j+1]) {
                  csere(&a[j],&a[j+1]);
                  cs=j;
            }
      }
      i=cs;
}
```

7. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
int i=6;
while (i>0) {
      int cs=-1;
      for (int j=0; j<i; j++) {
            if (a[j]>a[j+1]) {
                  csere(&a[j],&a[j+1]);
                  cs=j;
            }
      }
      i=cs;
}
```

24. melléklet: Beszúró rendezéssel kapcsolatos kérdőív (1. csoport - animáció)

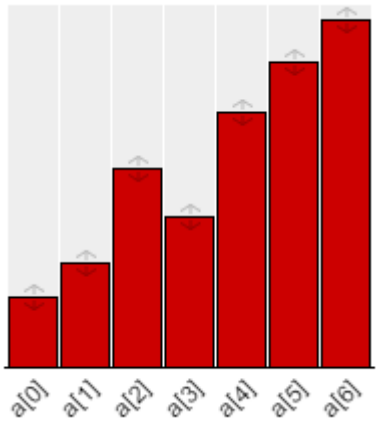
1. Az alábbiak közül mi igaz a **beszúró rendezésre (insertion sort)**? Több válasz is bejelölhető!

- ☐ egymástól távolabb elhelyezkedő elemeket is összehasonlít
- ☐ közvetlenül egymás melletti (szomszédos) elemeket hasonlít össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

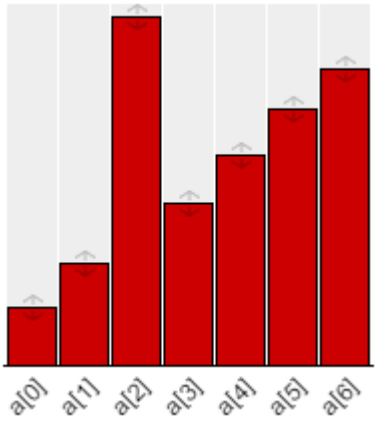
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** van amikor kisebb, van amikor nagyobb mint **j**
- ☐ az **i** van amikor kisebb, van amikor egyenlő **j**-vel, de sose nagyobb **j**-nél
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



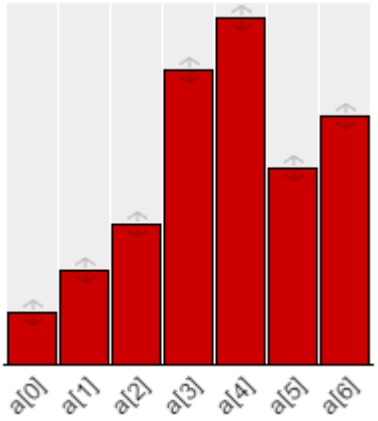
- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

4. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

5. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=1; i<=6; i++) {
     int j=i-1;
     while (j>=0 && a[j]>a[j+1]) {
          csere(&a[j],&a[j+1]);
          j--;
     }
}
```

7. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=1; i<=6; i++) {
     int j=i-1;
     while (j>=0 && a[j]>a[j+1]) {
          csere(&a[j],&a[j+1]);
          j--;
     }
}
```

25. melléklet: Beszúró rendezéssel kapcsolatos kérdőív (2. csoport - grafika)

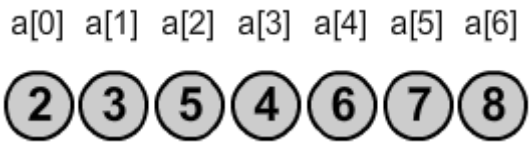
1. Az alábbiak közül mi igaz a **beszúró rendezésre (insertion sort)**? Több válasz is bejelölhető!

- ☐ egymástól távolabb elhelyezkedő elemeket is összehasonlít
- ☐ közvetlenül egymás melletti (szomszédos) elemeket hasonlít össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

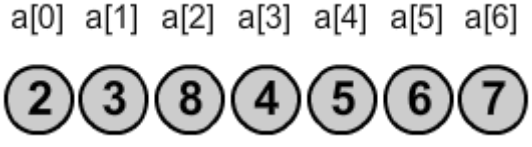
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** van amikor kisebb, van amikor nagyobb mint **j**
- ☐ az **i** van amikor kisebb, van amikor egyenlő **j**-vel, de sose nagyobb **j**-nél
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



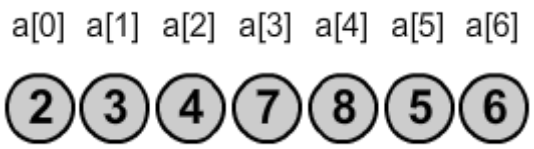
- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

4. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

5. Összesen mennyi cserét fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 7
- ☐ 9
- ☐ 10

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=1; i<=6; i++) {
     int j=i-1;
     while (j>=0 && a[j]>a[j+1]) {
          csere(&a[j],&a[j+1]);
          j--;
     }
}
```

7. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=1; i<=6; i++) {
     int j=i-1;
     while (j>=0 && a[j]>a[j+1]) {
          csere(&a[j],&a[j+1]);
          j--;
     }
}
```

26. melléklet: Továbbfejlesztett beszúró rendezéssel kapcsolatos kérdőív (1. csoport - animáció)

1. A **továbbfejlesztett beszúró rendezésre (improved insertion sort)** …

- ☐ több összehasonlítást végez mint az eredeti beszúró rendezés
- ☐ ugyanannyi összehasonlítást végez mint az eredeti beszúró rendezés
- ☐ kevesebb összehasonlítást végez mint az eredeti beszúró rendezés

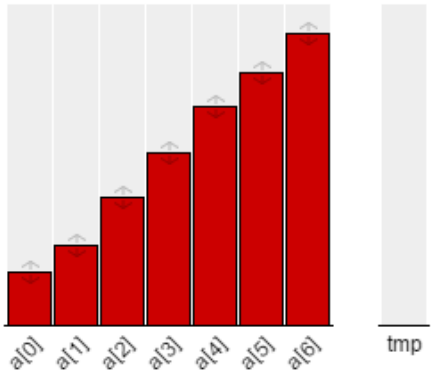
- ☐ több másolást végez mint az eredeti beszúró rendezés *
- ☐ ugyanannyi másolást végez mint az eredeti beszúró rendezés *
- ☐ kevesebb másolást végez mint az eredeti beszúró rendezés *

** az eredetiben egy csere 3 másolásnak felel meg*

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

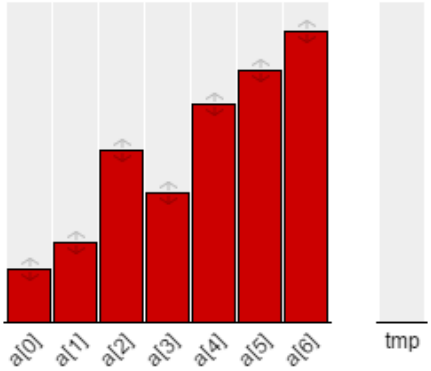
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** van amikor kisebb, van amikor nagyobb mint **j**
- ☐ az **i** van amikor kisebb, van amikor egyenlő **j**-vel, de sose nagyobb **j**-nél
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi másolást fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 6
- ☐ 7
- ☐ 12
- ☐ 13
- ☐ 14
- ☐ 15

4. Összesen mennyi másolást fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 6
- ☐ 7
- ☐ 12
- ☐ 13
- ☐ 14
- ☐ 15

5. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=1; i<=6; i++) {
     int j=i-1;
     int tmp=a[i];
     while (j>=0 && a[j]>tmp) {
          a[j+1]=a[j];
          j--;
     }
     a[j+1]=tmp;
}
```

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=1; i<=6; i++) {
     int j=i-1;
     int tmp=a[i];
     while (j>=0 && a[j]>tmp) {
          a[j+1]=a[j];
          j--;
     }
     a[j+1]=tmp;
}
```

27. melléklet: Továbbfejlesztett beszúró rendezéssel kapcsolatos kérdőív (2. csoport - grafika)

1. A **továbbfejlesztett beszúró rendezésre (improved insertion sort)** …

- ☐ több összehasonlítást végez mint az eredeti beszúró rendezés
- ☐ ugyanannyi összehasonlítást végez mint az eredeti beszúró rendezés
- ☐ kevesebb összehasonlítást végez mint az eredeti beszúró rendezés

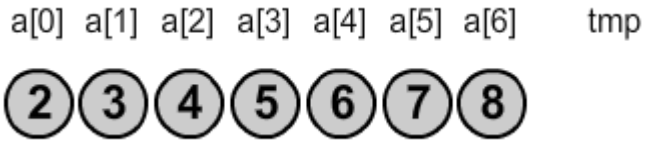
- ☐ több másolást végez mint az eredeti beszúró rendezés *
- ☐ ugyanannyi másolást végez mint az eredeti beszúró rendezés *
- ☐ kevesebb másolást végez mint az eredeti beszúró rendezés *

** az eredetiben egy csere 3 másolásnak felel meg*

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

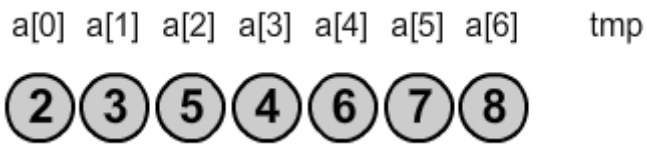
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** van amikor kisebb, van amikor nagyobb mint **j**
- ☐ az **i** van amikor kisebb, van amikor egyenlő **j**-vel, de sose nagyobb **j**-nél
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi másolást fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 6
- ☐ 7
- ☐ 12
- ☐ 13
- ☐ 14
- ☐ 15

4. Összesen mennyi másolást fog az algoritmus elvégezni, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 6
- ☐ 7
- ☐ 12
- ☐ 13
- ☐ 14
- ☐ 15

5. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=1; i<=6; i++) {
     int j=i-1;
     int tmp=a[i];
     while (j>=0 && a[j]>tmp) {
          a[j+1]=a[j];
          j--;
     }
     a[j+1]=tmp;
}
```

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=1; i<=6; i++) {
     int j=i-1;
     int tmp=a[i];
     while (j>=0 && a[j]>tmp) {
          a[j+1]=a[j];
          j--;
     }
     a[j+1]=tmp;
}
```

28. melléklet: Minimumkiválasztásos rendezéssel kapcsolatos kérdőív (1. csoport - animáció)

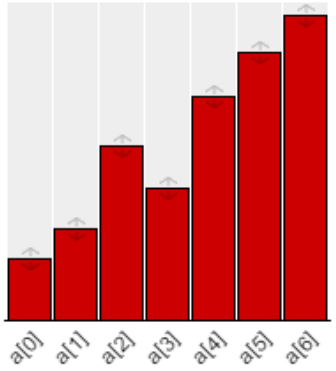
1. Az alábbiak közül mi igaz a **minimumkiválasztásos rendezésre (minsort)**? Több válasz is bejelölhető!

- ☐ egymástól távolabb elhelyezkedő elemeket is összehasonlít
- ☐ közvetlenül egymás melletti (szomszédos) elemeket hasonlít össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

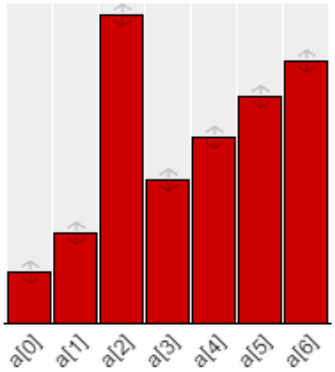
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig kisebb vagy egyenlő **j**-vel
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig nagyobb vagy egyenlő **j**-vel
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



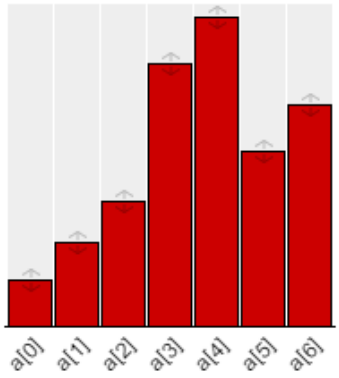
- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

4. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

5. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=0; i<6; i++) {
     int min=i;
     for (int j=i+1; j<=6; j++) {
          if (a[min]>a[j]) {
               min=j;
          }
     }
     csere(&a[i],&a[min]);
}
```

7. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=0; i<6; i++) {
     int min=i;
     for (int j=i+1; j<=6; j++) {
          if (a[min]>a[j]) {
               min=j;
          }
     }
     csere(&a[i],&a[min]);
}
```

29. melléklet: Minimumkiválasztásos rendezéssel kapcsolatos kérdőív (2. csoport - grafika)

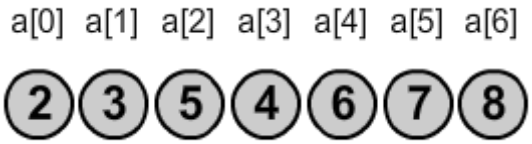
1. Az alábbiak közül mi igaz a **minimumkiválasztásos rendezésre (minsort)**? Több válasz is bejelölhető!

- ☐ egymástól távolabb elhelyezkedő elemeket is összehasonlít
- ☐ közvetlenül egymás melletti (szomszédos) elemeket hasonlít össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

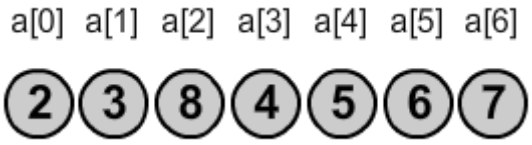
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig kisebb vagy egyenlő **j**-vel
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig nagyobb vagy egyenlő **j**-vel
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



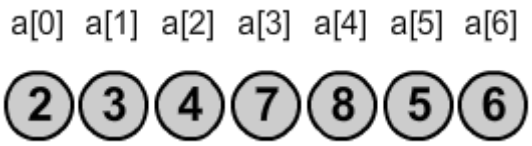
- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

4. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

5. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=0; i<6; i++) {
     int min=i;
     for (int j=i+1; j<=6; j++) {
          if (a[min]>a[j]) {
               min=j;
          }
     }
     csere(&a[i],&a[min]);
}
```

7. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=0; i<6; i++) {
     int min=i;
     for (int j=i+1; j<=6; j++) {
          if (a[min]>a[j]) {
               min=j;
          }
     }
     csere(&a[i],&a[min]);
}
```

30. melléklet: Maximumkiválasztásos rendezéssel kapcsolatos kérdőív (1. csoport - animáció)

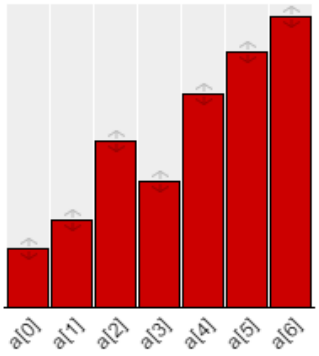
1. Az alábbiak közül mi igaz a **maximumkiválasztásos rendezésre (maxsort)**? Több válasz is bejelölhető!

- ☐ egymástól távolabb elhelyezkedő elemeket is összehasonlít
- ☐ közvetlenül egymás melletti (szomszédos) elemeket hasonlít össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

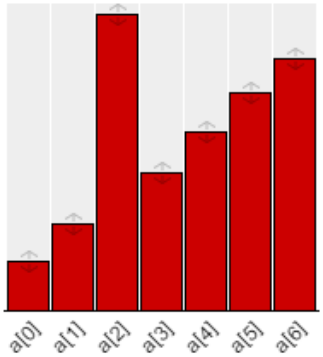
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig kisebb vagy egyenlő **j**-vel
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig nagyobb vagy egyenlő **j**-vel
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



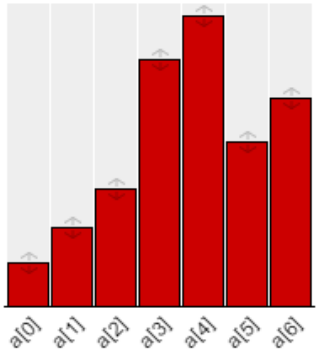
- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

4. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

5. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



☐ 0
☐ 1
☐ 2
☐ 3
☐ 4
☐ 5
☐ 6
☐ 7

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=6; i>0; i--) {
     int max=0;
     for (int j=1; j<=i; j++) {
          if (a[max]<a[j]) {
               max=j;
          }
     }
     csere(&a[max],&a[i]);
}
```

7. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=6; i>0; i--) {
     int max=0;
     for (int j=1; j<=i; j++) {
          if (a[max]<a[j]) {
               max=j;
          }
     }
     csere(&a[max],&a[i]);
}
```

31. melléklet: Maximumkiválasztásos rendezéssel kapcsolatos kérdőív (2. csoport - grafika)

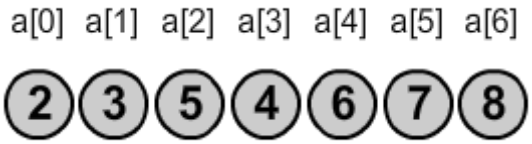
1. Az alábbiak közül mi igaz a **maximumkiválasztásos rendezésre (maxsort)**? Több válasz is bejelölhető!

- ☐ egymástól távolabb elhelyezkedő elemeket is összehasonlít
- ☐ közvetlenül egymás melletti (szomszédos) elemeket hasonlít össze
- ☐ a rendezett rész a tömb elejétől kezd kialakulni
- ☐ a rendezett rész a tömb végétől kezd kialakulni

2. Mi igaz az **i** és a **j** ciklusváltozókra a ciklusokon belüli feltétel vizsgálatakor? Csak egy válasz helyes! (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!)

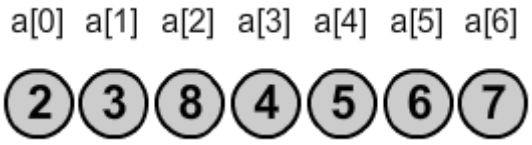
- ☐ az **i** mindig kisebb mint **j**
- ☐ az **i** mindig kisebb vagy egyenlő **j**-vel
- ☐ az **i** mindig nagyobb mint **j**
- ☐ az **i** mindig nagyobb vagy egyenlő **j**-vel
- ☐ az **i** mindig egyenlő **j**-vel
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



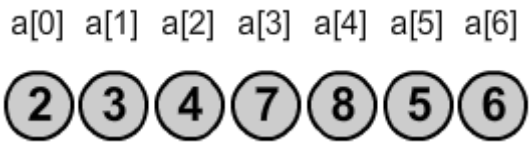
- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

4. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

5. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

6. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne 7-elemű, hanem N-elemű tömböt rendezzen!

```
for (int i=6; i>0; i--) {
     int max=0;
     for (int j=1; j<=i; j++) {
          if (a[max]<a[j]) {
               max=j;
          }
     }
     csere(&a[max],&a[i]);
}
```

7. Módosítsd az alábbi algoritmust (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a 7-elemű tömböt!

```
for (int i=6; i>0; i--) {
     int max=0;
     for (int j=1; j<=i; j++) {
          if (a[max]<a[j]) {
               max=j;
          }
     }
     csere(&a[max],&a[i]);
}
```

32. melléklet: Gyorsrendezéssel kapcsolatos kérdőív (1. csoport - animáció)

1. Az alábbiak közül mi igaz a **gyorsrendezésre (quicksort)**? Több válasz is bejelölhető!

- ☐ mindig egymás melletti elemeket cserél ki
- ☐ a tömbrész elején levő elemeket cseréli ki a hátul levő elemekkel

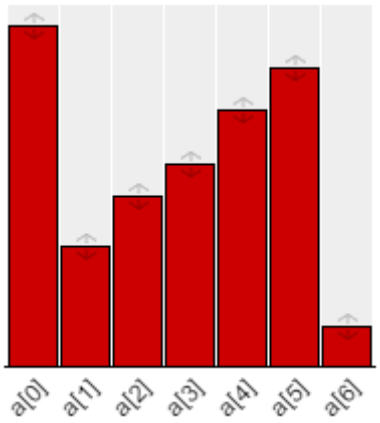
- ☐ csak egyszer megy végig a tömbön, közben elvégzi a szükséges cseréket
- ☐ többször is végigmegy a tömbön, mindig kisebb és kisebb tömbrészeken, amelyeken belül mindig elvégzi a szükséges cseréket

- ☐ a rendezett rész fokozatosan, a tömb elejétől hátrafele haladva kezd kialakulni
- ☐ a rendezett rész fokozatosan, a tömb végétől előre haladva kezd kialakulni
- ☐ a rendezett rész nem sorban alakul ki a tömb elejétől vagy a végétől, hanem akár a tömb közepén is a helyére kerülhet egy-egy elem a rendezés során

2. Mi igaz az **i** és a **j** változókra a **quicksort** függvény belsejében? (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!) Több válasz is bejelölhető!

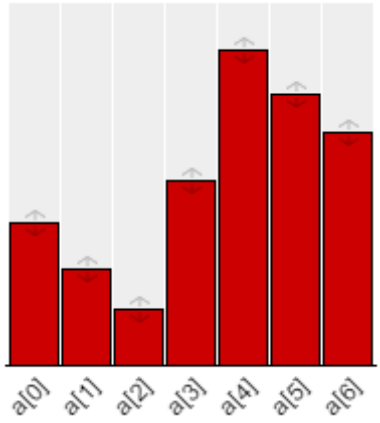
- ☐ az **i** mindig csökken
- ☐ az **i** mindig növekszik
- ☐ a **j** mindig csökken
- ☐ a **j** mindig növekszik
- ☐ az **i** mindig kisebb vagy egyenlő **j**-vel
- ☐ az **i** mindig nagyobb vagy egyenlő **j**-vel
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



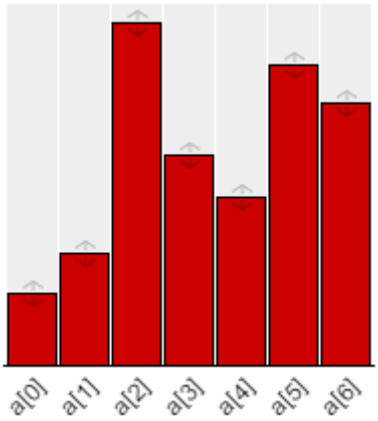
- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

4. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

5. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

6. Módosítsd az alábbi függvényt (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a tömböt!

```
void quicksort(int beg, int end) {
      int i = beg;
      int j = end;
      int pivot = a[(i+j)/2];
      while (i<=j) {
            while (a[i]<pivot)
                  { i++; }
            while (a[j]>pivot)
                  { j--; }
            if (i<=j) {
                  csere(&a[i],&a[j]);
                  i++;
                  j--;
            }
      }
      if (beg<j)
            { quicksort(beg,j); }
      if (i<end)
            { quicksort(i,end); }
}
```

33. melléklet: Gyorsrendezéssel kapcsolatos kérdőív (2. csoport - grafika)

1. Az alábbiak közül mi igaz a **gyorsrendezésre (quicksort)**? Több válasz is bejelölhető!

- ☐ mindig egymás melletti elemeket cserél ki
- ☐ a tömbrész elején levő elemeket cseréli ki a hátul levő elemekkel

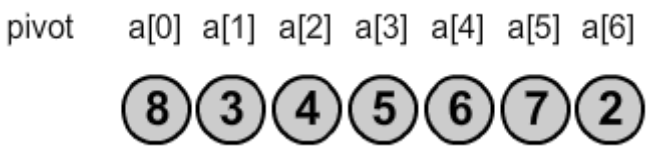
- ☐ csak egyszer megy végig a tömbön, közben elvégzi a szükséges cseréket
- ☐ többször is végigmegy a tömbön, mindig kisebb és kisebb tömbrészeken, amelyeken belül mindig elvégzi a szükséges cseréket

- ☐ a rendezett rész fokozatosan, a tömb elejétől hátrafele haladva kezd kialakulni
- ☐ a rendezett rész fokozatosan, a tömb végétől előre haladva kezd kialakulni
- ☐ a rendezett rész nem sorban alakul ki a tömb elejétől vagy a végétől, hanem akár a tömb közepén is a helyére kerülhet egy-egy elem a rendezés során

2. Mi igaz az **i** és a **j** változókra a **quicksort** függvény belsejében? (Figyelem, a kérdés az i és j változókra, nem az a[i] és a[j] változókra vonatkozik!) Több válasz is bejelölhető!

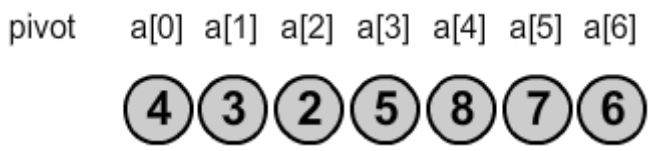
- ☐ az **i** mindig csökken
- ☐ az **i** mindig növekszik
- ☐ a **j** mindig csökken
- ☐ a **j** mindig növekszik
- ☐ az **i** mindig kisebb vagy egyenlő **j**-vel
- ☐ az **i** mindig nagyobb vagy egyenlő **j**-vel
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



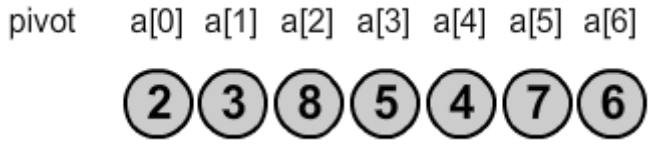
- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7

4. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



☐ 0
☐ 1
☐ 2
☐ 3
☐ 4
☐ 5
☐ 6
☐ 7

5. Összesen mennyi olyan cserét fog az algoritmus elvégezni, ahol az elemet nem önmagával cseréli ki, ha a rendezés előtt az alábbi tömbünk van?



☐ 0
☐ 1
☐ 2
☐ 3
☐ 4
☐ 5
☐ 6
☐ 7

6. Módosítsd az alábbi függvényt (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a tömböt!

```
void quicksort(int beg, int end) {
      int i = beg;
      int j = end;
      int pivot = a[(i+j)/2];
      while (i<=j) {
            while (a[i]<pivot)
                  { i++; }
            while (a[j]>pivot)
                  { j--; }
            if (i<=j) {
                  csere(&a[i],&a[j]);
                  i++;
                  j--;
            }
      }
      if (beg<j)
            { quicksort(beg,j); }
      if (i<end)
            { quicksort(i,end); }
}
```

34. melléklet: Összefésülő rendezéssel kapcsolatos kérdőív (mindkét csoport)

1. Az alábbiak közül mi igaz az **összefésülő rendezésre (mergesort)**?

- ☐ csak egyetlen egyszer megy végig a tömbön, közben elvégzi a rendezést
- ☐ többször is végigmegy a tömbön, mindig kisebb és kisebb tömbrészeken, amelyeken belül mindig elvégzi a tömbrészek rendezését

2. Mi igaz az **i** és a **j** változókra a **mergesort** függvény azon részében, ahol összefésüli az elemeket (a **mergesort** függvény utolsó **for** ciklusában)? Több válasz is bejelölhető!

- ☐ az **i** mindig csökken
- ☐ az **i** mindig növekszik
- ☐ a **j** mindig csökken
- ☐ a **j** mindig növekszik
- ☐ az **i** mindig kisebb vagy egyenlő **j**-vel
- ☐ az **i** mindig nagyobb vagy egyenlő **j**-vel
- ☐ az **i** lehet kisebb, nagyobb, vagy egyenlő is **j**-vel

3. A 8-elemű tömb esetében a **mergesort** rekurzív függvény 4 mélységig hívódott meg (tehát legtöbbször ennyiszer futott le a függvény egymáson belül). Milyen mélységig hívódna meg a függvény, ha…

16-elemű tömbünk lenne: …….

32-elemű tömbünk lenne: …….

2^{10}-elemű tömbünk lenne: …….

4. Összesen mennyi összefésülést (beleszámolva bármilyen rövid tömbész összefésülését) végzett el az algoritmus (a 8-elemű tömb teljes rendezéséhez)?

…….

5. Összesen mennyi összefésülést (beleszámolva bármilyen rövid tömbész összefésülését) végezne el az algoritmus, ha…

16-elemű tömbünk lenne: …….

32-elemű tömbünk lenne: …….

6. Módosítsd az alábbi függvényt (írd át az alábbi kinyomtatott programban azokat a részeket, amik megváltoznak) úgy, hogy az algoritmus ne növekvő, hanem csökkenő sorrendbe rendezze a tömböt!

```
void mergesort(int kez, int veg) {
     if (kez<veg) {
          int m = (kez+veg)/2;
          mergesort(kez,m);
          mergesort(m+1,veg);
          for (int i=kez; i<=m; i++)
                { x[i] = a[i]; }
          for (int i=m+1; i<=veg; i++)
                { int j = veg + m+1 - i;
                  x[j] = a[i]; }
          int i = kez;
          int j = veg;
          for (int k=kez; k<=veg; k++)
                if(x[i]<x[j])
                    { a[k] = x[i];
                      i++; }
                else
                    { a[k] = x[j];
                      j--; }
     }
}
```

35. melléklet: Rendezési algoritmusokkal kapcsolatos 16.-33. teszteken elért eredmények

Diák sorszáma	Csoport (1 - grafika, 2 - animáció)	Előtesztelés (%)	Egyszerű cserés rendezés (%) (simple exchange sort)	Buborékrendezés (%) (bubblesort)	Továbbfejlesztett buborékrendezés (%) (improved bubblesort)	Beszúró rendezés (%) (insertion sort)	Továbbfejlesztett beszúró rendezés (%) (improved insertion sort)	Minimumkiválasztásos rendezés (%) (selection sort: minsort)	Maximumkiválasztásos rendezés (%) (selection sort: maxsort)	Gyorsrendezés (%) (quicksort)	Összefésülő rendezés (%) (mergesort)
1.	1	67	88	100	88	100	100	89	100	91	82
2.	1	75	100	100	100	88	71	100	100	100	100
3.	1	75	63	100	100	88	71	100	100	91	100
4.	1	42	50	75	50	75	57	67	38	82	73
5.	1	58	75	88	75	88	57	78	75	73	91
6.	1	83	100	100	63	100	100	100	100	100	55
7.	1	33	75	88	75	88	43	67	88	100	36
8.	1	42	75	88	75	75	43	67	88	73	27
9.	1	33	50	63	75	75	57	67	88	73	55
10.	1	25	63	75	75	88	71	89	88	100	82
11.	1	50	75	63	63	75	29	56	75		82
12.	1	50	75	88	75	75	57	78	75	73	91
13.	1	42	75	88	50	75	57	78	75	73	91
14.	1	92	63	75	50	75	43	89	88	100	
15.	1	67	63	100	100	100	86	89	88	91	82
16.	1	58	100	100	100	100	100	100	100	100	55
17.	1	75	75	100	88	100	71	89	100	82	82
18.	1	67	75	88	75	88	71	89	88	82	73
19.	1	83	63	100	75	100	57	78	100	82	91
20.	1	58	63	100	75	100	57	100	100	100	73
21.	1	83	75	100	88	100	86	100	100	100	73
22.	1	67	88	100	100	100	71	100	88	73	
23.	1	25	63	63				67	63	73	45
24.	1	100	75	88	88	75	43	100	100	100	
25.	1	100	100	88	100	100	100			82	73
26.	1	92	75	88	75	88	86	100	100	55	64
27.	1	42	75	88	63	63	71	67	75	82	45
28.	1	50	38	50	63	63	57	78	50		
29.	1	25	63	100				78	75		
30.	1	42	100	100	75	100	71	100	88	91	91
31.	1	83	75	100	100	100	100	89	100	91	55
32.	1	58	75	100							
33.	1	8	50	63	50	75	43	44	50	45	

34.	1	75	75	88	88	88	71	89	75	55	64
35.	1										36
36.	2										55
37.	2	75	75	88	75	100	71	89	75	64	55
38.	2	83	75	75	88	88	86	44	100	73	
39.	2	58	50	88	38	88	43	67	88	100	64
40.	2	58	63	63	25	75	29	56	50	73	36
41.	2	33	50	88	63	75	29	78	75	73	82
42.	2	50	50	75	50	88	71	89	88	64	55
43.	2	50	25	38	13	50	14	33	63	82	64
44.	2				38	88	29			64	
45.	2				63	88	71	56	75	73	
46.	2	75	63	88	38	88	57			64	
47.	2	33	50	50	38	88	71	56	75	73	36
48.	2	67	63	100	63	75	71			82	
49.	2	25	63	75	13	50	0	22	25		
50.	2	50	38	100	50	75	29	89	100		
51.	2	75	75	88	75	88	86	89	75		
52.	2	92	88	88	63	88	57	89	100		82
53.	2	42	75	63	38	63	43	89	75		36
54.	2	67	50	75						36	
55.	2	17	50	75	38	75	0	67	50	73	
56.	2	25	50	75	25	63	0			73	27
57.	2	33	38	50	0	50	14	56	50	73	18
58.	2	33	63	88	75	88	43	67	88		27
59.	2	92	100	100	75	100	86	89	88	100	100
60.	2	100	88	100	100	100	71	100	88	100	64
61.	2	25	75	88	63	88	100	78	100	55	
62.	2	33	63	88	38	88	43			73	
63.	2	33	50	75	50	75	29	56	88	45	
64.	2	42	50	63	50	75	43			73	55
65.	2	83	63	100	100	88	57	89	88		73
66.	2	33	63	88	63	75	43			55	27
67.	2	75	100	100	100	100	86	100	100		
68.	2	33	63	75	38	75	43	44	75		
69.	2	50	50	88	25	88	57	89	88		
70.	2	25	63	63	38	75	43	44	75		
71.	2	92	88	100	63	100	29	89	88		27

[1]**ADATLAP**
a doktori értekezés nyilvánosságra hozatalához

I. A doktori értekezés adatai

A szerző neve: Végh Ladislav
MTMT-azonosító: 10034170
A doktori értekezés címe és alcíme: A programozás tanulásának és tanításának támogatása elektronikus tananyagba beépíthető interaktív animációs modellekkel
DOI-azonosító[2]: 10.15476/ELTE.2017.124
A doktori iskola neve: Eötvös Loránd Tudományegyetem Informatika Doktori Iskola
A doktori iskolán belüli doktori program neve: Az informatika alapjai és módszertana
A témavezető neve és tudományos fokozata: Prof. Ing. Stoffa Veronika, CSc., egyetemi professzor
A témavezető munkahelye: Nagyszombati Egyetem, Tanárképző Kar, Trnava, Szlovákia
A témavezető neve és tudományos fokozata: Dr. Turcsányiné Szabó Márta, egyetemi docens
A témavezető munkahelye: Eötvös Loránd Tudományegyetem, Informatikai Kar, Budapest

II. Nyilatkozatok

1. A doktori értekezés szerzőjeként[3]

a) hozzájárulok, hogy a doktori fokozat megszerzését követően a doktori értekezésem és a tézisek nyilvánosságra kerüljenek az ELTE Digitális Intézményi Tudástárban. Felhatalmazom az Informatika Doktori Iskola hivatalának ügyintézőjét, Boda Annamáriát, hogy az értekezést és a téziseket feltöltse az ELTE Digitális Intézményi Tudástárba, és ennek során kitöltse a feltöltéshez szükséges nyilatkozatokat.

b) kérem, hogy a mellékelt kérelemben részletezett szabadalmi, illetőleg oltalmi bejelentés közzétételéig a doktori értekezést ne bocsássák nyilvánosságra az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban;[4]

c) kérem, hogy a nemzetbiztonsági okból minősített adatot tartalmazó doktori értekezést a minősítés (*dátum*)-ig tartó időtartama alatt ne bocsássák nyilvánosságra az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban;[5]

d) kérem, hogy a mű kiadására vonatkozó mellékelt kiadó szerződésre tekintettel a doktori értekezést a könyv megjelenéséig ne bocsássák nyilvánosságra az Egyetemi Könyvtárban, és az ELTE Digitális Intézményi Tudástárban csak a könyv bibliográfiai adatait tegyék közzé. Ha a könyv a fokozatszerzést követőn egy évig nem jelenik meg, hozzájárulok, hogy a doktori értekezésem és a tézisek nyilvánosságra kerüljenek az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban.[6]

2. A doktori értekezés szerzőjeként kijelentem, hogy

a) az ELTE Digitális Intézményi Tudástárba feltöltendő doktori értekezés és a tézisek saját eredeti, önálló szellemi munkám és legjobb tudomásom szerint nem sértem vele senki szerzői jogait;

b) a doktori értekezés és a tézisek nyomtatott változatai és az elektronikus adathordozón benyújtott tartalmak (szöveg és ábrák) mindenben megegyeznek.

3. A doktori értekezés szerzőjeként hozzájárulok a doktori értekezés és a tézisek szövegének plágiumkereső adatbázisba helyezéséhez és plágiumellenőrző vizsgálatok lefuttatásához.

Kelt: 2017.8.23.

a doktori értekezés szerzőjének aláírása

[1] Beiktatta az Egyetemi Doktori Szabályzat módosításáról szóló CXXXIX/2014. (VI. 30.) Szen. sz. határozat. Hatályos: 2014. VII.1. napjától.
[2] A kari hivatal ügyintézője tölti ki.
[3] A megfelelő szöveg aláhúzandó.
[4] A doktori értekezés benyújtásával egyidejűleg be kell adni a tudományági doktori tanácshoz a szabadalmi, illetőleg oltalmi bejelentést tanúsító okiratot és a nyilvánosságra hozatal elhalasztása iránti kérelmet.
[5] A doktori értekezés benyújtásával egyidejűleg be kell nyújtani a minősített adatra vonatkozó közokiratot.
[6] A doktori értekezés benyújtásával egyidejűleg be kell nyújtani a mű kiadásáról szóló kiadói szerződést.